Krypton Help Pages on

# DMM Modular

Version 2.4

**May 2003**

Research Park Haasrode
Z1 Interleuvenlaan 86
3001 Leuven Belgium
tel.: +32 (0)16 400 410
fax: +32 (0)16 400 386
e-mail: info@krypton.be
http://www.krypton.be

KRYPTON
INDUSTRIAL METROLOGY

# Table of contents

1. START HERE ..... 11

2. INTRODUCTION ..... 12

3. INSTALLING THE DMM MODULAR SOFTWARE ..... 14

4. PRINCIPLES OF DMM MODULAR ..... 17

4.1. Measuring off-line ..... 17

4.2. Measuring single points (3D) ..... 17

4.3. Measuring Frames (6D) ..... 17

4.4. Transform to a user-chosen reference frame ..... 18

4.5. Orientation conventions ..... 19

5. MEASUREMENT SETUP CONSIDERATIONS ..... 20

5.1. Environment considerations ..... 20

5.2. Camera position & setting considerations ..... 20

6. ACTIONS BEFORE CONDUCTING MEASUREMENTS ..... 21

6.1. Configuring the camera's ..... 21

6.2. Checking LED visibility & environment ..... 21

6.3. Checking the Probe ..... 21

6.4. Calibrating the Probe ..... 21

7. USING DMM MODULAR ..... 22

7.1. The DMM Modular user interface ..... 22

7.1.1. The main menu ..... 23

7.1.2. The workspace window ..... 23

7.1.3. The output window ..... 24

7.2. Configuring DMM Modular ..... 24

8. IDENTIFYING OBJECTS ..... 25

9. CONFIGURING A MEASUREMENT ..... 26

**9.1. Tab 'General'.....26**

**9.2. Tab 'camera'.....27**
9.2.1. 9.2.1 Configuring LED's for single point measurements.....29
9.2.2. Configuring LED's for a dynamic frame.....29

**9.3. Tab 'Analog'.....30**

**9.4. Tab 'digital'.....31**

**9.5. Tab 'Counters'.....31**

**10. MEASURING SINGLE LED'S OR OBJECTS.....32**

**11. CALCULATING RESULT FILES.....34**

**11.1. Calculation module 'Acquisition'.....34**
11.1.1. The function 'measure'.....34
11.1.2. The function 'Create 6D data'.....34
11.1.3. The function 'Transform'.....35

**11.2. Calculation module 'Base'.....35**
11.2.1. The function 'Offset'.....36
11.2.2. The function 'Distance'.....37
11.2.3. The function 'Calculate velocity'.....37
11.2.4. The function 'overshoot'.....38
11.2.5. The function 'statistics'.....39
*11.2.5.1 The function 'Find maximum' "*.....39
*11.2.5.2 The function 'Find minimum'*.....39
*11.2.5.3 The function 'Calculate mean'*.....39
*11.2.5.4 The function 'Calculate std'*.....40
11.2.6. The function 'export'.....40
11.2.7. The function 'Change orientation convention'.....40
11.2.8. The function 'Calculate point in 6D frame'.....41
11.2.9. The function 'fit polynome'.....42
11.2.10. The function 'Filter'.....44
11.2.11. The function 'Frequency'.....44
11.2.12. The function 'Join measurements'.....45

**11.3. Adding Calculation modules.....46**

**12. CREATING SPREADSHEETS & GRAPH'S.....47**

**12.1. Working with cursors.....47**
12.1.1. Working with single cursors.....47
12.1.2. Working with difference cursors.....49
12.1.3. Remaining cursor options.....49

**12.2. The spreadsheet view.....49**
12.2.1. The 'Data' menu.....50
12.2.2. The 'Edit' menu.....51
12.2.3. The 'Search' menu.....51
*12.2.3.1 Criterium*.....51
*12.2.3.2 Reset search*.....52

**12.3. The 2D-graph ..... 52**

**12.4. The 3D-view ..... 55**

*12.4.1.1 Cursor controls ..... 56*

*12.4.1.2 Animation controls ..... 56*

*12.4.1.3 View controls ..... 57*

**13. EXPORTING TO MATLAB ..... 59**

**14. OPTIONS IN THE MAIN MENU ..... 60**

**14.1. 'Project' menu ..... 60**

**14.2. 'View' menu ..... 62**

**14.3. 'Tools' menu ..... 62**

14.3.1. 'Options' ..... 62

*14.3.1.1 General options ..... 63*

*14.3.1.2 ASCII Export options ..... 63*

*14.3.1.3 Acquisition options ..... 64*

14.3.2. 'Customize' ..... 65

14.3.3. 'Users' ..... 66

**14.4. 'Identify' menu ..... 67**

**14.5. 'Script' menu ..... 67**

**14.6. 'Acquisition' menu ..... 69**

**14.7. 'Base' menu ..... 70**

**14.8. 'Help' menu ..... 70**

**15. SCRIPTING – WRITING AND USING MACRO'S ..... 71**

**15.1. Writing a single-level script ..... 71**

15.1.1. Writing a script ..... 71

15.1.2. The script syntax ..... 72

**15.2. Using scripts within scripts ..... 73**

**16. SCRIPT FUNCTION REFERENCE ..... 75**

**16.1. acquisition( ) ..... 77**

**16.2. addfromfile( ) ..... 77**

**16.3. calculatemean( ) ..... 78**

**16.4. calculatestd( ) ..... 78**

**16.5. clearmemory( ) ..... 78**

16.6. compensatelinearphasefilter( ) ..... 79
16.7. copy( ) ..... 79
16.8. createframe( ) ..... 79
16.9. delete( ) ..... 79
16.10. distance( ) ..... 80
16.11. error( ) ..... 80
16.12. export( ) ..... 81
16.13. FFT( ) ..... 81
16.14. filecopy( ) ..... 82
16.15. filter( ) ..... 82
16.16. findmaximum( ) ..... 82
16.17. findminimum( ) ..... 83
16.18. globaladd( ) ..... 83
16.19. globalemptylist( ) ..... 83
16.20. identify( ) ..... 83
16.21. increasetestname( ) ..... 84
16.22. joinmeasurement( ) ..... 84
16.23. message( ) ..... 84
16.24. offset( ) ..... 85
16.25. orientationchange( ) ..... 85
16.26. overshoot( ) ..... 86
16.27. pointinframe( ) ..... 86
16.28. polyfit( ) ..... 87
16.29. rename( ) ..... 87
16.30. saveall( ) ..... 87
16.31. settestname( ) ..... 88
16.32. terminate( ) ..... 88
16.33. transform( ) ..... 88

16.34. velocity( ) ..... 89
16.35. wait( ) ..... 89
16.36. waitmessage( ) ..... 89
16.37. warning( ) ..... 89

17. DATA ORGANIZATION ..... 90
17.1. The ‘test’ level ..... 90
17.2. Global variables ..... 91

18. MAINTENANCE ..... 93

19. TROUBLESHOOTING ..... 94
19.1. General ..... 94
19.2. Identifying objects ..... 94
19.3. Measuring single LED’s or objects ..... 94
19.4. Adding calculation modules ..... 94
19.5. Creating spreadsheets & graph’s ..... 94
19.6. Exporting to Matlab ..... 95

20. GLOSSARY ..... 96

21. INDEX ..... 97

# TABLE OF FIGURES

Figure 1 : release CD - pop-up form ....14
Figure 2: LED & camera ....17
Figure 3: dynamic frame with 3 LED's ....18
Figure 4: frame transformation ....18
*Figure 5: DMM Modular - user interface* ....22
Figure 6: Identifying objects ....25
Figure 7: the measurement configuration pop-up form ....26
Figure 8: measurement configuration - tab 'camera' ....28
*Figure 9: measurement configuration - tab 'analog'* ....30
Figure 10: measurement configuration - tab 'digital' ....31
Figure 11: measurement configuration - tab 'counters' ....31
Figure 12: acquisition – status window ....32
Figure 13: 'create 6D variable' pop-up form ....34
Figure 14: 'Transform' pop-up form ....35
*Figure 15: 'Offset' pop-up form* ....36
Figure 16: the 'distance' pop-up form ....37
*Figure 17: 'Velocity' pop-up form* ....38
Figure 18: the 'overshoot' pop-up form ....38
Figure 19: 'file export' pop-up form ....40
Figure 20: 'change orientation convention' pop-up form ....41
Figure 21: 'calculate point in 6D frame' pop-up form ....41
Figure 22: the 'polynomial fit' pop-up form ....43
Figure 23: polynomial fit - the polynomial coefficients ....43
Figure 24: the 'filter' pop-up form ....44
Figure 25: the 'frequency' pop-up form ....45
Figure 26: the 'join measurements' pop-up form ....46
Figure 27: cursor tab in output window ....47
Figure 28: cursor value selector forms ....48
Figure 29: cursor column headings ....48
Figure 30: difference cursors ....49
Figure 31: opening a spreadsheet view ....50
Figure 32: the spreadsheet view ....50
Figure 33: cursor annotations ....50
Figure 34: opening a 2D-graph ....52
*Figure 35: 2D-graph variables selection pop-up form* ....53
Figure 36: 2D-graph legend window ....54
Figure 37: opening a spreadsheet view ....55
Figure 38: 3D view settings window – general settings ....57
Figure 39: 3D view settings window – general settings ....58
Figure 40: 3D view settings window – 3D connect lines settings ....58
*Figure 41: 'New project' window* ....61
Figure 42: General settings tab sheet ....63
*Figure 43: 'ASCII export' settings tab sheet* ....63
Figure 44: 'Acquisition' settings tab sheet ....64
Figure 45: 'Customize' tab sheet - toolbars ....65
Figure 46: 'Customize' tab sheet - keyboard ....66
*Figure 47: script editor - the function browser* ....69
Figure 48: Scripting - recording a script ....72
*Figure 49: Scripting - editing a script* ....72
Figure 50: script parameter pop-up form ....74
Figure 51: data organisation: object-based (left) or person-based (right) ....90
Figure 52: data organisation: 'set test name' ....91

Figure 53: data organisation: incrementing test names from a script .....91

# Product protection

No copies of this manual or of parts of it may be made in any form or by any means without the written permission of KRYPTON N.V.
©Copyright 2002 KRYPTON N.V.

The software and hardware described in these KRYPTON Help Pages are furnished under a license agreement and may be used only in accordance with the terms of this agreement.

The contents of the KRYPTON Help Pages is strictly confidential and the material described is strict intellectual and industrial property of KRYPTON N.V.. Divulgation of any kind of information described in this manual is strictly forbidden.

No warranty is given on damages, including but not limited to loss of profit, and special, incidental, consequential, or other similar claims.

ROCAL, Rodym, GeoLoc and SPACE PROBE are registered trademarks of KRYPTON N.V.
Microsoft and Windows are registered trademarks of Microsoft Corporation.

# 1. Start here

Welcome to the DMM Modular user manual.

DMM Modular is a software package to measure and analyze complex kinematic and dynamic properties of structures, both in 3D and 6D. Each acquisition can be expressed in a static or dynamic user-specified reference coordinate system.

This guide is intended for all users working with DMM Modular. This document will guide you through all the features this software has to offer.
This guide will follow a top-down approach, as you would approach the system for the first time. Step by step, we'll go through all of the possibilities, starting with the obvious features, and ending with the most powerfull and versatile features.

This guide assumes you have a basic knowledge of the Krypton camera systems, the GeoLoc software, CMM 5.xx and that you have some basic experience with the Windows operating system.
This guide assumes that all hardware is set up, and that all components (including the Space® Probe) are operational.

If you don't have a basic knowledge of the Krypton camera systems, or you are interested in its features, please refer to the following manuals :

- "*Rodym6D system guide*" and "*Rodym6D utilities guide*" if you are using a Rodym camera
- "*Krypton help pages on Kx00® Utilities"* if you are using a Kx00 camera.

If you don't have a basic knowledge of the Krypton GeoLoc software, or you are interested in its features, please refer to the manual "*GeoLoc user guide*"

If you are interested in the features of DMM Modular, start with chapter "Introduction" on page 12.

If you want to know more about the measurement principles of DMM Modular, start with chapter "Principles of DMM Modular" on page 17.

If you want to install the software from scratch, start with chapter "Installing the DMM Modular software" on page 14.

If you want to review the general considerations for dynamic measurements, start with chapter "Measurement setup considerations" on page 20.

If you want to make sure that everything is set before the first measurement takes place, start with chapter "Actions before conducting measurements" on page 21.

If you are sure that all preliminary actions are taken, or were not required, start with chapter "Using DMM Modular" on page 22.

If you want to look up certain terms or words that are not clear to you, refer to the Glossary on page 96, or use the Index on page 97.

# 2. Introduction

Krypton develops and commercialises camera-based solutions for measuring and evaluating the 6D movements of objects in space. Our systems are based on non-contact measurements using solid state camera systems. Several software packages are developed by Krypton to support industrial and laboratory position problems, both static and dynamic, but always using the same hardware.

Each camera system consists of 1 or more digital infrared CCD camera's mounted on a fixed beam. Each of the camera's measures an angle of an infrared LED that resides in the camera's field-of-view. By combining the readings from the three camera's, a 3D-position of the LED can be calculated with respect to the camera's reference frame.
If we attach 3 such LED's on an object, we can calculate the 6D-motion of this object by combining the 3D-readings of the three LED's.

DMM Modular is a software package that measures mechanical vibrations and motions.

Before the measurement, DMM Modular determines the relation between an object and LED's that are attached to the object. During the measurement, the software measures the LED positions in the camera coordinate system. After the measurement, DMM Modular calculates the actual motion of the object, based on the positions of the LED's on the object.

Finally, a "transformation" can be performed to express the motion of the object no longer in the camera coordinate system, but in a reference coordinate system that is chosen by the user. This reference frame can be static or dynamic.

DMM Modular uses GeoLoc to identify objects; GeoLoc also identifies the frames that will act as reference frames.

DMM Modular allows the user to show his measurement in a spreadsheet, a 2D-graph, and even a 3D-animation.
The spreadsheet allows to search for specific data points (minima, maxima, ...) .
The 2D-graph allows zooming, numeric zooming, panning, …
The 3D-view gives a clear insight in the motion of the different objects.

The measured data can be exported to Matlab with one click, allowing the user to freely experiment with measured data.

DMM Modular allows additional calculation modules to be inserted quite easy. If a customer has a Matlab calculation routine, Krypton can convert this into a compiled module. The software will recognise this module and shortcuts to these modules will be inserted in the menu and icon bar of the application.

"Scripting" allows the user to teach DMM Modular a measurement sequence once, and replaying it many times. This prevents the user from repeated, annoying keyboard manipulations. Load a script, run it – and DMM Modular will handle all needed acquisitions and calculations.

DMM Modular can be extended with an additional acquisition box, the ADC600 : this box will measure analog, digital and encoder/counter signals, perfectly synchronous with the camera, allowing a perfect match of position information with force measurements, shaft positions, …

The software runs under Windows 98 or higher versions. In order to reduce the calculation time, a Pentium III 500 MHz processor or better is advised. Free hard disk space requires a minimum of 160 Mbyte, and a minimum of 128 Mbyte RAM memory.

The system comes with a CD-ROM containing the installation version of DMM Modular.

This page is only intended for users that are upgrading to a newer version.

**Software changes from version 2.2 to version 2.4 :**

- From now on, DMM Modular comes in two versions, 'Standard' and 'Advanced'. The 'Advanced' version supports exportation to Matlab, and has more functions. See chapter "Calculation module 'Base'" on page 35.
- The DMM Modular now supports 2 Krypton camera families: the Rodym-camera and the K600-camera.
- The 3D view has been changed: from now on, it supports *.DXF-files, LED's in the 3D view can be connected by lines, a "Copy to Clipboard" function, Z-axis invertion, … . See chapter "The 3D-view" on page 55.
- The function "Transform" has been modified. See chapter "The function 'Transform'" on page 35.
- The function "Change orientation convention" has been modified. See chapter "The function 'Change orientation convention'" on page 40.
- The function "export" has been extended with an RPC3 output format. See chapter "The function 'export'" on page 40.
- When exporting to file from within a script, script variables can be used to replace fixed test names or sequence numbers. See chapter "export( )" on page 81.
- New functions have been added to the base module: filter, frequency, and joinmeasurements. See chapter "Calculation module 'Base'" on page 35.

**Software changes from version 2.1 to version 2.2 :**

- When creating 6D frames, the data file sequence number will be equal to the number of the corresponding led files. Untill now, this numbering was independent and could lead to confusion.
- All the icons have tooltips, to get a quick indication of their function
- The spreadsheet has a "Delete" function, that allows to trim non-functional data from a data file. See chapter "The 'Edit' menu" on page 51.
- 2D-graphs have a new ruler that indicates dX, dY and distance between 2 points. See chapter "The 2D-graph" on page 52.
- scripts will no longer output messages to a separate tab sheet in the output window
- when recording a script, variable names are put between double quotes automatically
- Scripts can run from a specific line, without commenting the previous lines: see chapter "'Script' menu" on page 67.
- Geoloc is closed automatically when DMM needs the camera.
- New functions have been added to the base module: minimum, maximum, average, standard deviation, and overshoot. See chapter "Calculation module 'Base'" on page 35.
- The memory management has been improved: recent files remain in memory to accelerate calculations, older files are removed to free up memory.

**Software changes from version 2.0 to version 2.1 :**

- The acquisition pop-up form was slightly changed. See chapter "Measuring single LED's or objects" on page 32.
- A new function has been added to allow a polynomial fit on raw data. See chapter "The function 'fit polynome' " on page 42.
- The spreadsheet underwent minor changes.
- The 'Customize' menu was changed :see chapter "'Options'" on page 62.
- When recording a script, it is now possible to append to an existing script.
- The organisation of data in projects has been extended, and is fully explained in a separate chapter: see "Data organisation" on page 90.

# 3. Installing the DMM Modular software

Power up your PC and close all unnecessary applications.

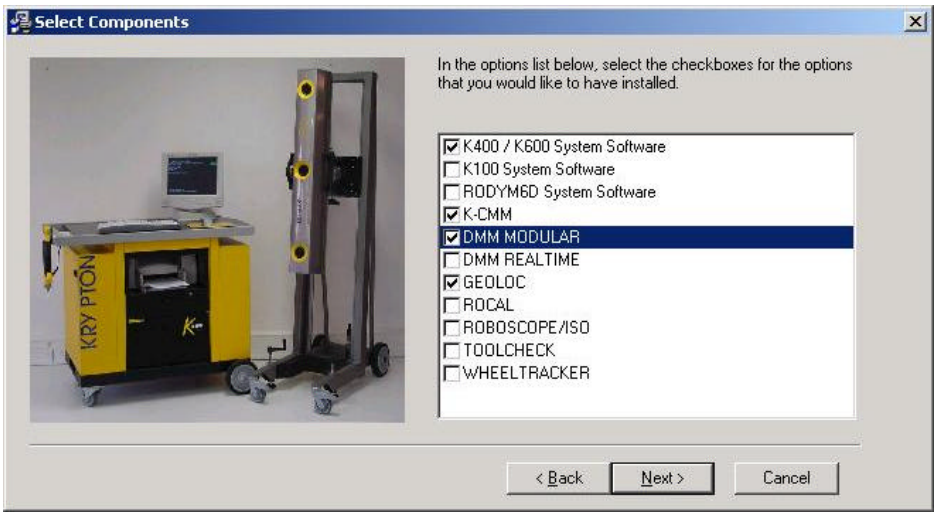
Insert the Krypton installation CD in your CD-ROM drive. The following screen – release date may be more recent – will appear automatically:

Welcome
Welcome to Krypton Software Products Setup program. This program will install Krypton Software Products on your computer.
It is strongly recommended that you exit all Windows programs before running this Setup Program.
Click Cancel to quit Setup and close any programs you have running. Click Next to continue with the Setup program .
WARNING: This program is protected by copyright law and international treaties.
Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.
Next >
Cancel

*Figure 1 : release CD - pop-up form*

If you didn't get this screen automatically within 30 seconds, use your Windows Explorer to locate the file *SETUP.EXE* on the CD; start this by double-clicking on the file.

Click on the button *"Next"*. A list of software products will appear, sorted per product group:



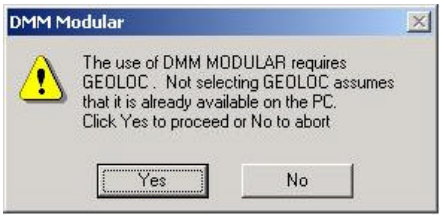

Check the following software packages if they were not already installed on your system; however, we advise you to install these software packages in advance. Please refer to the user manuals of the software packages involved :

- "K400/K600 system software" (if you have a Kx00 camera system) or "Rodym6D System Software" (if you have a Rodym6D camera system)
- "K-CMM"

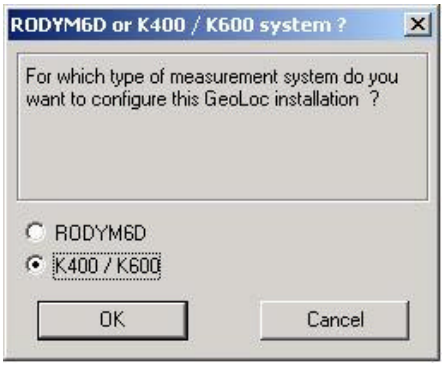
?? “GeoLoc”

Check “DMM Modular” and click “Next”.

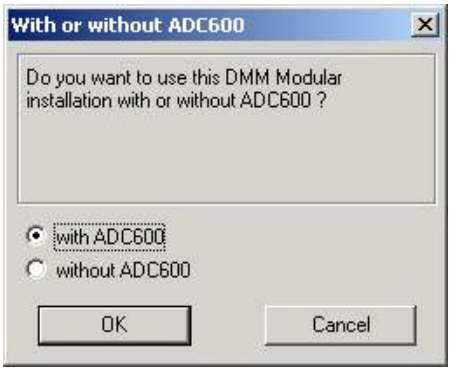
If you did not select GeoLoc, the installation program application will warn you of the requirement to have GeoLoc installed previously. Clicking “Yes” will continue the installation of DMM, “No” will abort the installation process; in the latter case, restart and install the required programs before DMM Modular.


DMM Modular
The use of DMM MODULAR requires GEOLOC . Not selecting GEOLOC assumes that it is already available on the PC.
Click Yes to proceed or No to abort
Yes
No

In the welcome screen, read the instructions as shown on the screen; keep on clicking the “Next” button untill you reach the following screen:

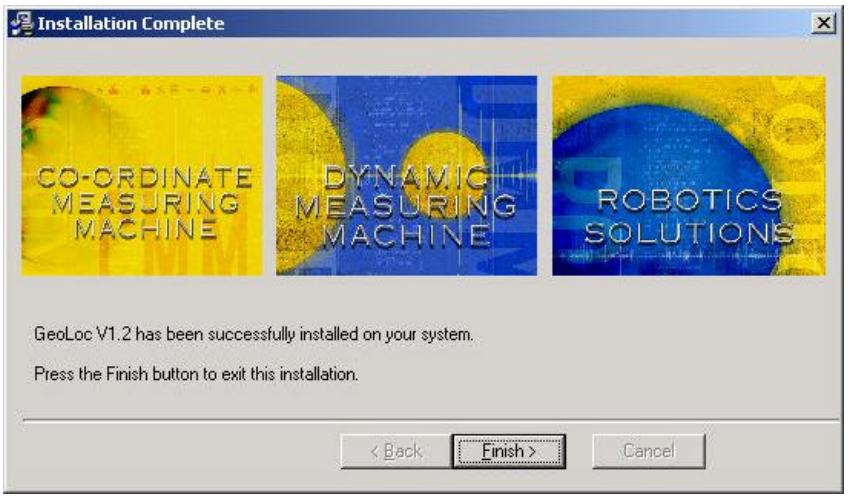

RODYM6D or K400 / K600 system ?
For which type of measurement system do you want to configure this GeoLoc installation ?
RODYM6D
K400 / K600
OK
Cancel

Select the camera type you currently use ( Kx00 or Rodym6D) and click “OK”. This takes you to the next screen:


With or without ADC600
Do you want to use this DMM Modular installation with or without ADC600 ?
with ADC600
without ADC600
OK
Cancel

Choose wether you have a system with the optional ADC600. If the ADC600 is supplied and connected, you **must** choose “with ADC600”. After selecting, click the “OK” button. Before continuing, you get a reminder to install a driver from a separate CD :

Click the “OK” button; follow the instructions on the screen, and keep on clicking “Next”, untill you reach the following screen :


Installation Complete
CO-ORDINATE MEASURING MACHINE
DYNAMIC MEASURING MACHINE
ROBOTICS SOLUTIONS
GeoLoc V1.2 has been successfully installed on your system.
Press the Finish button to exit this installation.
< Back
Finish >
Cancel

Click the “Finish” button to conclude the DMM Modular installation proc ess.
If the main installation program is still active, click the “Finish” button of that program to conclude the entire installation process.

# 4. Principles of DMM Modular

When using DMM Modular, it is usefull to consider the main philosophy behind the software.

The main idea is :

“To measure off-line single point-motions (3D) or frame-motions (6D) in camera coordinate system, and transform these data into a new user-chosen reference frame“

The following sub-chapters will clarify some of the key-terms :

## 4.1. Measuring off-line

When measuring off-line, all data acquired by the camera is first saved in a temporary buffer. No calculation whatsoever is done during the measurement. The advantage is, that all resources are available to acquire data. All calculations are performed when the measurement is terminated. **Measuring off-line results in the highest acquisition frequency.**

Krypton also has a RealTime package: in RealTime, a single snapshot (or “frame”) is measured, calculated, and outputted in a user-chosen form (analog or network output). The maximal real-time acquisition frequency is 300 Hz.

## 4.2. Measuring single points (3D)

For some measurement setups, you are only interested in 3 translations of a point. The three remaining degrees of freedom (3 rotations) remain undetermined.

In this case, a single LED is attached to an object. The camera will measure the X, Y and Z-coordinate of the LED.

The advantage of measuring single points is the ease-of-use : you can start directly measuring. Interpreting the results is mostly rather straight-forward.

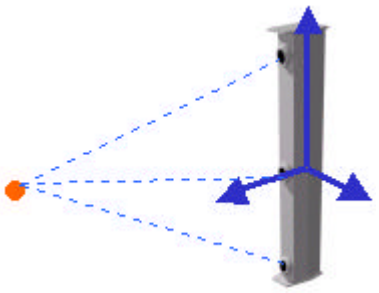


*Figure 2: LED & camera*

The camera measures the optical center of the LED, so you can not pin-point the position of the point-of-interest within 5 mm of accuracy.

## 4.3. Measuring Frames (6D)

If the point-of-interest must be pin-pointed within 0.1 mm , or if you are also interested in the rotational degrees of freedom, you must measure at least three LED’s on an object to eliminate the three undetermined degrees of freedom.

The LED’s can have arbitrary positions in the object frame. The 3 translations and 3 rotations describe the position and orientation of the frame located in the origin of the object frame.

The following figure illustrates the principle of measuring LED’s and calculating frame positions : if we know the position of each LED’s in the object’s frame in rest, and we can

measure the individual LED's in a motion, we can calculate the new position of the object's frame, based on the nominal LED positions. Such a frame with LED's is called a dynamic frame .

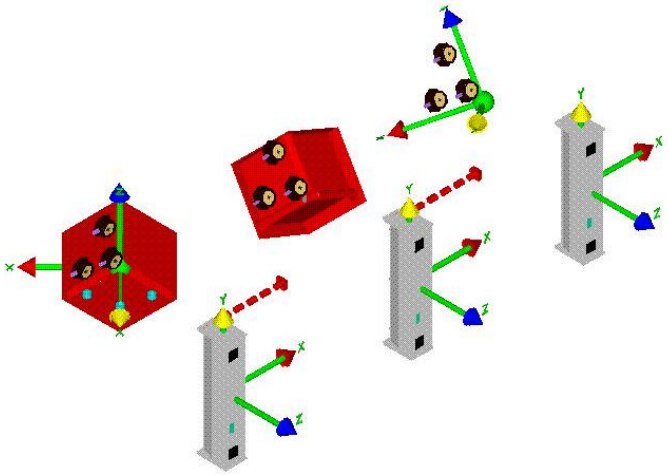


*Figure 3: dynamic frame with 3 LED's*

How to calculate the LED positions in object frame:

1) Identify the position and orientation of the object frame with respect to the camera. This is a static measurement in which we determine the relation between camera frame and object frame.
2) Attach the LED's to the object and measure the position of these LED's in camera frame.
3) You can now transform the LED positions expressed in camera frame into object frame. This transformation was established in step 1)

The advantage of measuring frames is the ability to pin-point the point-of-interest, namely the origin of a dynamic frame, and the ability to view the rotations for complex motions. A third advantage is the possibility to create multiple dynamic frames with 1 set of LED's : this way, we can calculate multiple points of interest with only one set of LED's.

Using frames requires the additional step of identifying a dynamic frame, and a slightly longer calculation time.

## 4.4. Transform to a user-chosen reference frame

Wether we measured both single points or frames, both of them are initially expressed in the camera reference frame. In many cases, the user wants to express the measurements in another frame.

Therefore, we choose to use a measured frame as a new reference. We know the transformation between this frame and the camera frame (green arrow) . Next, we use this transformation to transform the single points or frames expressed in camera frame (blue arrow) into the user-chosen reference frame (red).

The frame, chosen to act as reference frame, can be either static or dynamic.

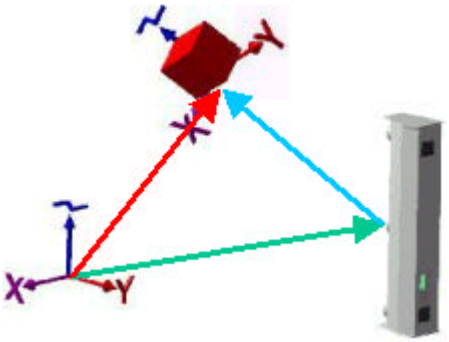


*Figure 4: frame transformation*

## 4.5. Orientation conventions

When executing translations, the order of translation is of no importance.
This is NOT the case with rotations: e.g. when we rotate object A 10° around X, 20° around Y and 30° around Z, and we rotate object B 30° around Z, 20° around Y and 10° around X, then the orientations of the two objects will differ.

This has a reverse effect: if an object holds a certain orientation, it is possible to achieve that orientation with more than 1 sequence of rotations around the axes – and consequently, the angles will be different. In some cases, a mathematical solution even corresponds to a physical condition that is not possible (e.g. the steering of a wheel exceeding 100° )

There is also the question on 'absolute rotations' or 'relative rotations' : suppose an object, which is aligned with a reference frame. If the object is rotated around its X-axis, the object's Y- and Z-axis rotated over the same angle. But if I want to rotate around Y, will I rotate around the Y-axis that followed the rotation (relative rotation), or will I rotate around an axis, parallel to the Y-axis of the reference frame (absolute rotation) ?

Therefore, an "orientation convention" must be chosen: such a convention enforces a certain sequence of rotation around the axes of the reference frame, and imposes whether the rotation will be absolute or relative.
It is up to you to determine which convention suits your needs best.

Once an orientation convention is chosen, and the software returns three rotation angles, the following rule must be observed:
"**The object's orientation in space is the same as when executing the rotation angles by the sequence and constraints of the orientation convention**".

DMM Modular supports three types of orientation conventions :

**Roll-Pitch-Yaw** : rotation order is X-Y-Z. The rotations are absolute. Preferred convention for naval applications ( for a ship the following conventions apply: longitudinal axis:X - lateral axis:Y - vertical axis:Z). The three coordinates are the roll-angle, the pitch-angle and the yaw-angle.
**Steer-Camber-Spin** : rotation order is Z-X-Y. The rotations are relative. Preferred convention for automotive applications ( for a car the following conventions apply: longitudinal axis:X - lateral axis:Y - vertical axis:Z). The three coordinates are the steer-angle, the camber-angle and the spin-angle.
**Screw vector** : a convention in which we construct a vector in a 3D-space: rotating around this vector brings us with 1 rotation from position A to B. The angle of rotation is the vector norm (in radians) – this is always the smallest angle to transit. Preferred convention when indicating small orientation errors (comparison expected-measured path). Disadvantage is an ambiguity once the vector norm exceeds ?. The three coordinates are the lengths of the three vectors $S_x$, $S_y$ and $S_z$ that compose the screw vector.

When using the first two orientations, it is important to avoid identifications that place a frame axis 1 or 3 parallel to the reference frame axis 3 or 1: this is a "singular position", a position where it is possible to calculate an indefinite number of points. Such situations will amplify noise and reduce accuracy of the angles. In general, if an object frame is identified parallel to the reference frame, there is no problem.

# 5. Measurement setup considerations

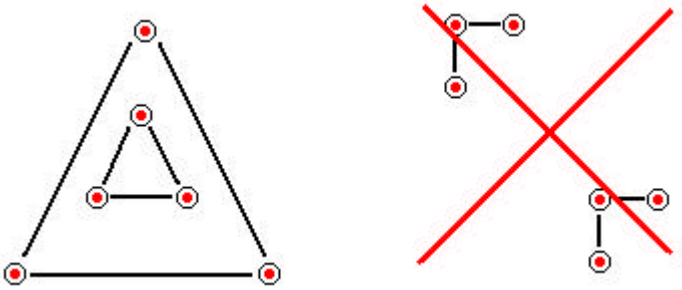
This chapter describes a few precautions to take when using a Krypton camera system to optimise visibility and accuracy of a measurement.
Naturally, these considerations can be overruled by factors imposed by the measurement setup: reference points that are not easy to acces, LED adhesion points, ...

## 5.1. Environment considerations

- Avoid hot light sources (containing infrared light), by not orienting the camera towards windows, lights, sun,…
- Avoid air turbulences. Place the camera horizontal to reduce the influence. Close all doors in the room. Switch off heating devices and fans where possible.
- Avoid to place LED's in such places where reflective surfaces can reflect the infrared light to the camera
- If EMC problems arise, use an insulative medium between the LED's and the object(s). Use a shielding around the strober cable, in case problems persist.
- **NEVER** mix equipment from two camera families! Always use the Space Probe, LED's and strobers supplied with a camera. Do not use a Space Probe from another camera. Do not use LED's or strobers from a different camera family (Rodym, Kx00, …)

## 5.2. Camera position & setting considerations

- Put the camera as close as possible to the LED's. Place the LED's as close as possible to the central axis of the camera
- Orient the LED's perpendicular to the camera. There is little accuracy degradation for angles of +/- 30°!
- Try to measure the primary motion in the camera XY plane: X and Y are the most accurate axes of the camera.
- In general, use a high LED frequency. For accurate measurements at larger distances, use lower LED frequencies.
  - If you are using a Rodym camera, use CamCheck to specify a LED frequency of 3250 Hz. See the manual "*Rodym6D system guide*".
  - If you are using a Kx00 camera, use K-Check to specify a marker frequency of 3250 Hz. See the manual "*Krypton help pages on Kx00 utilities".*
- Avoid using daisy-chained strober cables where possible, especially with high measurement frequencies.
- Avoid to identify object frames with an orientation of 90° (or close to this value) with respect to the reference frame (see chapter "Orientation conventions" on page 19).
- When using a 6D external reference, always try to put the external LED's on an equal-side triangle outside the 6D frame, maximizing distance between LED's.



# 6. Actions before conducting measurements

Before measuring, a few steps must be taken. Although they are completely independent from the DMM Modular software, DMM Modular assumes that these steps were carried out before the measurements start. Therefore, please make sure to perform these actions before continuing.

In sequence, these steps are :
1) configuring the camera's
2) checking LED visibility and the quality of the measurement environment
3) calibrating and checking the Space Probe

The following steps must be repeated for each camera PC on the test bench.

## 6.1. Configuring the camera's

If you purchased a Kx00® camera, this step is no longer required.

If you purchased a Rodym® 6D camera, refer to "*Rodym6D system guide*" , chapter "*configuring & operating the Rodym® 6D hardware*".

## 6.2. Checking LED visibility & environment

If you purchased a Kx00® camera, refer to the manual "Krypton help pages on Kx00® Utilities", chapter "K-Check: LED visibility & environment".

If you purchased a Rodym® 6D camera, refer to "*Rodym6D system guide*" , chapter "*configuring & operating the Rodym® 6D hardware*" .

## 6.3. Checking the Probe

If you purchased a Kx00® camera, refer to the manual "Krypton help pages on Kx00® Utilities", chapter "ProbeCheck: checking the Probe functionality".

If you purchased a Rodym® 6D camera, refer to "*Rodym6D utilities user guide*" , chapter "*using the Rodym® 6D utilities*"

## 6.4. Calibrating the Probe

If you purchased a Kx00® camera, refer to the manual "Krypton help pages on Kx00® Utilities", chapter "ProbeCal: calibrating the Probe".

If you purchased a Rodym® 6D camera, refer to "*Rodym6D utilities user guide*" , chapter "*using the Rodym® 6D utilities*"

# 7. Using DMM Modular

The following chapters will describe all options and features of DMM Modular.

When measuring with Rodym DMM Modular in general, always follow these steps :

1) Identification of the object(s)
2) measuring the LED's
3) calculations and analysis of the object(s) motions
4) showing the results

First, we'll take a closer look into the DMM Modular user interface, so we know which information can be found or expected where on the screen. Next, we'll discuss how to measure 3D- and 6D-motions, and how to represent them. Finally, we'll discuss the possibilities of the main menu. Each of the 4 steps will be explained in a separate chapter.

This chapter is written assuming that all default values for directories, program groups and such were chosen during installation.

## 7.1. The DMM Modular user interface

In this chapter, we'll look at the DMM Modular user interface, and the division into different zones. The use-and-yield of each zone is independent from the chronology of operations and is therefore discussed as first topic.

Open the program group "*Krypton*". Start "*RodymDMM*":

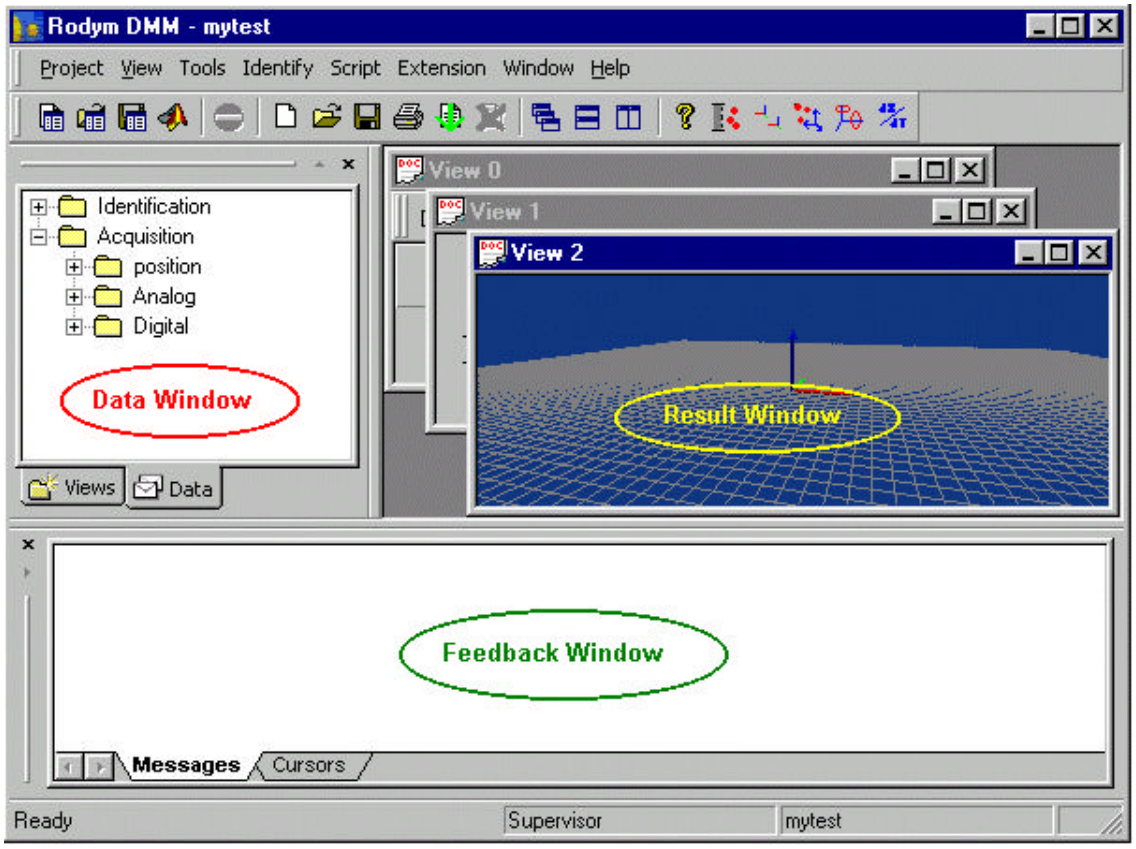


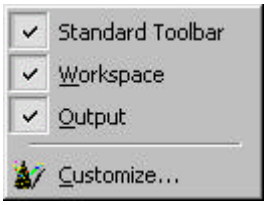

*Figure 5: DMM Modular - user interface*

You will be prompted to enter a user login and a password. By default, the user login is "supervisor" and the password is nil (no password). New users can be created, and their rights can be modified ( see chapter " 'Users' " on page 66.

The interface is divided into 4 different zones, each with a specific function :

1. **main menu / icon bar** on top. Through this menu, all options, configurations and features are accessible. Each option, configuration or feature corresponds to an icon that acts as a shortcut. Each of these icons can be dragged onto the icon bar or can be removed.
2. "**Workspace window**" (red) on the left: this window is the reservoir in which all measured data files are displayed in the "Data" - tab. In the "View" – tab, a new data view can be selected.
3. "**Result window**" (yellow) on the right: in this window, each new graph, spreadsheet or 3D-view will appear.
4. the "**Feedback window**" (green) at the bottom: this window holds all feedback from the program to the user. The window has 3 tabs :
    a. Messages: these are general messages, warnings or error reports from DMM Modular
    b. Cursors : if you add a cursor, the measurement values will be displayed in this tab.
    c. Commands : this window echoes all the actions the user has taken untill now.

If you accidentally closed the output window, move the mouse over the main menu, and right-click with the mouse : a menu will appear: check the appropriate window to be displayed again




### 7.1.1. The main menu

The main menu / icon bar is discussed in chapter "Options in the main menu" on page 60.

### 7.1.2. The workspace window

The workspace window will list all measured and calculated data files. Every file will be shown in a directory-like file organization :

**Identification:** contains all object identifications ( see chapter "Identifying objects" on page 25.
**Acquisition:** contains all raw acquisition files:
**Camera:** raw individual LED 3D data.
**Position3D:** individual LED 3D data in world reference frame
**Analog:** ADC600 analog input measurements
**Digital:** ADC600 digital input measurements
**Counters:** ADC600 encoder/counter input measurements
**Position6D:** contains calculated object 6D motion files

Later, these files can be dragged to the result window and can be dropped on either a spreadsheet, a 2D-graph or a 3D-view.

If you select a file, and right-click on it, a small menu pops up.
If you select "Properties", a two-tab properties sheet appears :
Base properties : administrative data on the file
3D properties: this option allows you to link a 3D object file to the frame in the data file. This object file contains a CAD description of the object who's motion is described in the

frame data file. If you open a 3D-view later, the object will be drawn around the frame, and you'll see the object move as it did during the measurement.

### 7.1.3. The output window

The feedback window will hold all messages from DMM Modular for the user. These messages can be an informational message ( e.g. reporting the residual error after a best-fit ), a warning message ( e.g. inaccurate calculation) or errors (e.g. "an error occurred in the module 'BASELOC' ").

The messages will appear in different tab sheets, depending on the context. DMM Modular itself will activate the proper tab for you when a new message arrives.

## 7.2. Configuring DMM Modular

DMM Modular expects you to make a few configuration settings before measuring. These settings should be checked :

1. the very first time DMM Modular is started,
2. If configuration changes are required.

It is a good practice to check these settings before measuring anyway. For a detailed explanation of the settings, go to chapter "'Options'" on page 62.

Go to the menu "Tools" – "Options". Open the tab "General" and check :

- if the default orientation convention is correct,
- if you want to use Matlab in DMM Modular scripts,
- if the Geoloc application path is filled in.

Open the tab "ASCII export" and check :

- if the data export ASCII file extension is correct,
- if the row delimiter is correct,
- if the decimal separator character is correct.

Open the tab "Acquisition" and check :

- if the CamCheck path text box is filled in,
- if the maximum allowed deformation is set to the requested value,
- if the ADC600 checkbox is checked or unchecked, depending on the presence and usage of the ADC600.

# 8. Identifying objects

If you are measuring frames, or if you want to express objects in a reference frame other than the camera, then this is the first step of a measurement session. If you are merely measuring single LED's in camera frame, proceed with chapter "Configuring a measurement" on page 26.

Should you encounter a problem, please refer to the troubleshooting chapter "Identifying objects" on page 94.

If you'd like, you could create a new project (projects will be discussed in chapter "Options in the main menu" on page 60) .

**Make sure that during the entire identification the object and camera do NOT move with respect to each other**: during the identification, an alignment will be created that becomes invalid once either the object or the camera move.

Open in the main menu the option "Identify" – "Start" .
This will start GeoLoc. GeoLoc will be opened with a project that is imposed by DMM Modular.

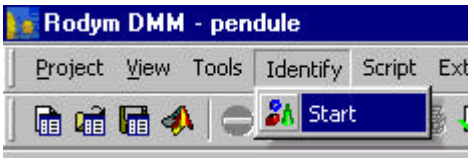



*Figure 6: Identifying objects*

Use GeoLoc to create a dynamic frame for every object of which you want to track the 6D-motion :

- mount the object in a fixed position with respect to the camera
- mount at least three LED's on the object rigidly. Connect the LED's to a strober, and the strober to a strober port ( see Rodym® 6D system guide )
- measure with GeoLoc the coordinate system of the object as if you are measuring a static frame ( see GeoLoc manual ) : let's call this frame <*object*_rest> ( replace object by your chosen name )
- create with GeoLoc a dynamic frame ( see GeoLoc manual ). Name the dynamic frame <*object*>. Use the static frame <*object*_rest> as external coordinate system.
- Repeat this for every object you want to track in 6D.

During these actions, bear in mind the recommendations specified in chapter "Measurement setup considerations" on page 20.

Drag all the created dynamic frames into GeoLoc's operand window and "teleport" the dynamic frames (see GeoLoc manual) to DMM Modular. They will appear in DMM Modular in the "identification" folder of the workspace window.

# 9. Configuring a measurement

This sub-chapter describes how to measure raw LED data.
This LED data can be viewed as movements of single points (3D), or can be grouped to derive an object's 6D-motion from.

An acquisition is started by choosing "Acquisition" – "Measure" from the main menu, or by clicking the Measure-icon on the icon-bar

A pop-up form will appear with 5 tab sheets. If you have installed an ADC600, the last 3 will be enabled; if not, the last three will be disabled.

## 9.1. Tab 'General'

This tab configures all common settings for the acquisition.

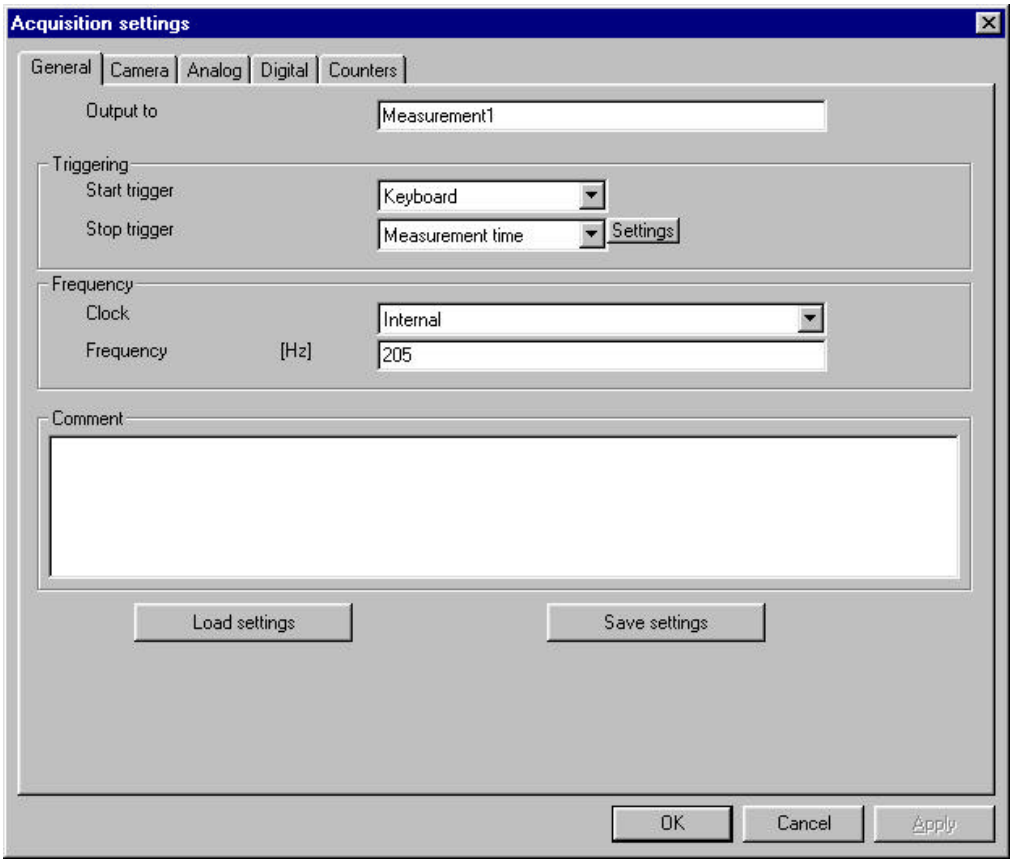



*Figure 7: the measurement configuration pop-up form*

**Output to:** name of the measurement file. It will be filed in the sub-folder "Position3D" of the folder "Acquisition" in the workspace window.

**Start trigger:** configures the start of the measurement.

- "user" : measurement will start by hitting the keyboard, or clicking with the mouse
- "external" : acquisition will trigger on a "true" TLL-signal, either on the ADC600, or directly on the camera control unit.
- "immediate" : the acquisition will start immediate, without a user acknowledgement.

**Stop trigger:** configures how to halt a measurement.
- “user” : measurement will halt by hitting the keyboard, or clicking with the mouse
- “external” : acquisition will stop on a “false” TLL-signal on the ADC600.
- “Measurement time” : acquisition will stop after a specified amount of time. Click the “settings”-button to specify the measurement time interval.

**Clock:** configures where to get the acquisition clock from:
- “Internal” : the system will use its own internally generated clock
- “external” : the system will use an external TTL clock.

**Frequency:** the acquisition frequency.

**Comment:** (optional) comment field

**Save settings:** allows to store the settings on the hard disk. Settings will be stored in a “*.settings”-file.

**Save settings:** allows to retrieve previously stored settings from a “*.settings”-file.

**Important notes :**
- When working without an ADC600, selecting the stop trigger “External TTL” will not work. The start trigger “External TTL” is available. See the manual “*Rodym6D system guide*” to learn more about the external synchronisation.
- When working with an ADC600, the stop trigger “External TTL” will only work if the start trigger is “External TTL” too.
- When using an external clock signal, with an ADC600 , the clock signal should be applied on the ADC600 “ext CLK in” input ( see the ADC600 documentation to learn more about external synchronisation). Samples are taken on a “false”-“true” edge ( this is a rising/falling edge, depending on wether your ADC600 is configured with active-high/active-low clock input ).
- When using external synchronisation without an ADC600, the clock signal should be applied directly on the camera control unit ( see the camera manual to learn more about external synchronisation). Samples are taken on a “false”-“true” edge ( this is a rising/falling edge, depending on your camera type ).
- When using an external clock signal, make sure the applied signal does not differ more than 2% from the frequency as configured in the tab-sheet “General”.

## 9.2. Tab ‘camera’

The tab ‘camera’ configures the acquisition of LED positions. First, we’ll explain the general features. Next, we’ll discuss how to configure single LED’s or dynamic frames to measure.

The general options are :

**Enable** : if checked, the camera will measure LED’s. The only case where you shouldn’t check this box is when you want to measure only with the ADC600.

**World** : the reference frame: all LED’s and dynamic frames will be expressed relative to the frame entered in this field. *The reference frame can be either a static or a dynamic frame !* If the field remains empty, or the frame does not exist, the reference frame will be the camera frame. The measurements will be expressed in this frame automatically after the acquisition.

**Back up 3D camera data** : if checked, the software will make a backup of all the individual LED’s in camera frame; should one of the calculated data files get corrupt, the data can be recalculated based on this raw data. The backup's are stored in the workspace window under "Acquisition-camera"

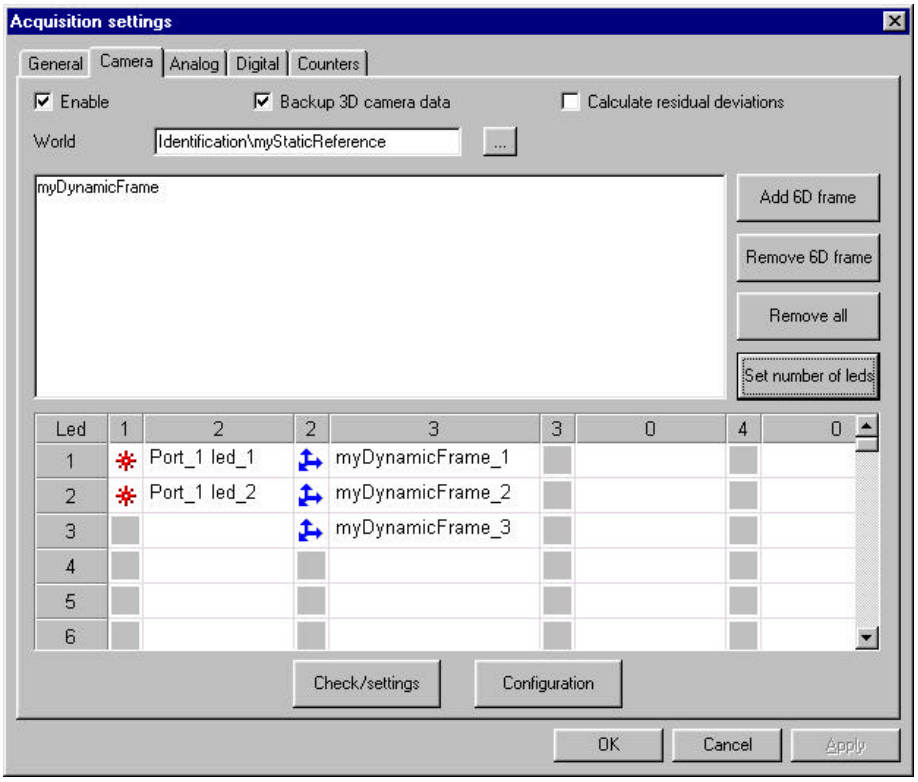



*Figure 8: measurement configuration - tab 'camera'*

**Calculate residual deviations** : if checked, for each frame the residual deviation will be calculated. This is a good quality check: if the residual deviation gets too big, there was an error calculating the frame position from the LED's, and the frame's 6D position is therefore not reliable. If you chose to calculate the residual errors, a time history of the residual errors is calculated and is accessible like any other data file.

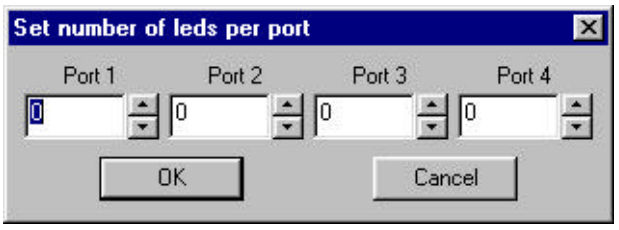
**Check settings** : will start CamCheck or an equivalent application. See your camera system guide.

**Configuration** : configures the camera. See chapter "Actions before conducting measurements" on page 21.

Before configuring your LED's, please review the recommendations specified in chapter "Measurement setup considerations" on page 20.

### 9.2.1. 9.2.1 Configuring LED's for single point measurements

To configure LED's for 3D-measurements, click on the button labeled "Set number of LED's".




In the pop-up form, configure for each strober the number of LED's attached to that port (including LED's on daisy-chained strobers ).

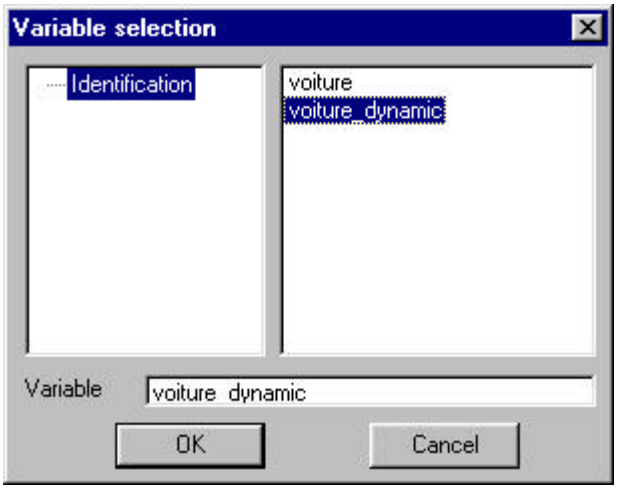
Click "OK", each LED will be listed as a red star in the overview. Each LED has a default name, being its port number and LED number.
You can change this default name by double-clicking on the LED name and overtyping it with a new name.

To remove single LED's, use the same form to reduce the number of LED's for a given strober port.

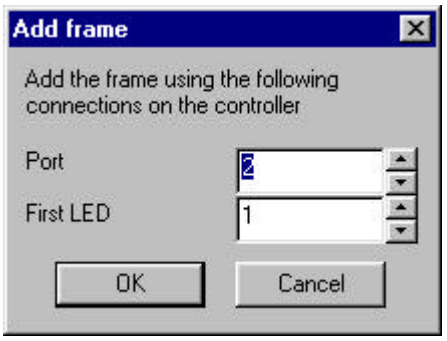
### 9.2.2. Configuring LED's for a dynamic frame

To configure the LED's of a dynamic frame, don't configure the LED's individually; just add the dynamic frame as it was defined.

Click on the button 'Add 6D frame', and the following pop-up will appear:




Select the dynamic frame from the folder "Identification" and confirm the choice.




Configure on which strober port the LED's of the object are attached, and which is the first LED.
**Note:** You don't have to specify the number of LED's on the frame, the software knows this from the definition of the dynamic frame.

The first LED is the sequence number of the first LED on the object. If you connect a strober on port 2, LED's 1 and 2 are single LED's, and we have a dynamic frame on LED's 3, 4 and 5, then "first LED" must be "3".

**Make sure the LED's have same sequence ( first LED, second LED, third … ) as during the object identification !**

Click “OK”, each LED is listed as a blue frame in the overview. Each LED has a default name, being its frame name extended with the LED sequence number in the frame definition.
You can change this default name by double-clicking on the LED name and overtyping it with a new name.

All chosen dynamic frames are listed in the big text field in the middle of the screen.
To remove a frame, select the frame until it is high-lighted and click “Remove 6D frame”. The frame will be removed from the text field. **Its LED’s remain configured, albeit as single LED’s !**

If frames are configured, the software will automatically calculate the frame motion from the motions of the frame’s single LED’s.

## 9.3. Tab ‘Analog’

This tab is only relevant if you have an ADC600 for measuring analog inputs.
If you have no ADC600, proceed with chapter “Measuring single LED’s or objects” on page 32.
To enable or disable all 16 channels at once, check/uncheck the “Enable” check-box on top.

Configure for each channel the following:

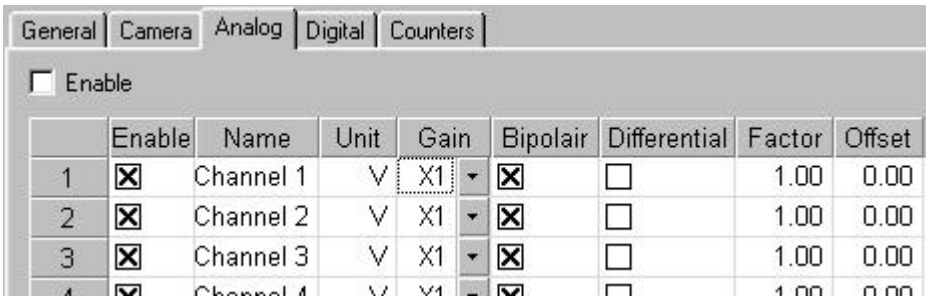


*Figure 9: measurement configuration - tab 'analog'*

**Enable**: enable/disable the individual channel

**Name**: keep the default name, or specify your own channel name by double-clicking and overtyping the name.

**Unit**: specify the unit you want the result to be expressed in

**Gain**: select the amplification factor of the channel. Use only one of the factors you can select from the combo box.
**Bipolar**: uncheck to only allow unipolar signals ( but with a doubled resolution ). Remember that unipolar signals need a gain of at least x2 !

**Differential**: not implemented yet

You can have the result immediately as a linear function of the voltage: Y = M * X + Q
**Factor**: “Factor” is the “M”, the linear scale factor. ***This scale factor should NOT be altered when changing the gain !*** The gain is applied only to the analog signal. Once the analog value is digitized, the numerical value is divided again by the same gain factor. Therefore, the scale factor is always applied to a value as if the gain was X1.

**Offset** “Offset” is the “Q” in the linear scaling

## 9.4. Tab ‘digital’

This tab is only relevant if you have an ADC600 for measuring digital inputs.
If you have no ADC600, proceed with chapter “Measuring single LED’s or objects” on page 32.
If the “Enable” checkbox on top is unchecked, all 32 channels will be disabled. If the checkbox is checked, individual channels must be checked for acquisition.

Configure for each channel the following:

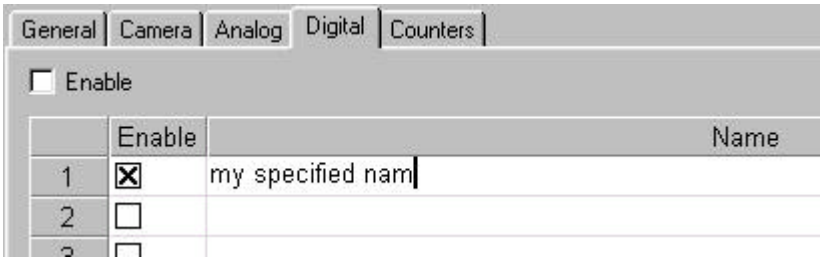



*Figure 10: measurement configuration - tab 'digital'*

**Enable**: enable/disable the individual channel

**Name**: keep the default name, or specify your own channel name by double-clicking and overtyping the name.

## 9.5. Tab ‘Counters’

This tab is only relevant if you have an ADC600 for measuring counter inputs.
If you have no ADC600, proceed with chapter “Measuring single LED’s or objects” on page 32 .
To enable or disable all 3 channels at once, check/uncheck the “Enable checkbox” on top.
Configure for each channel the following:

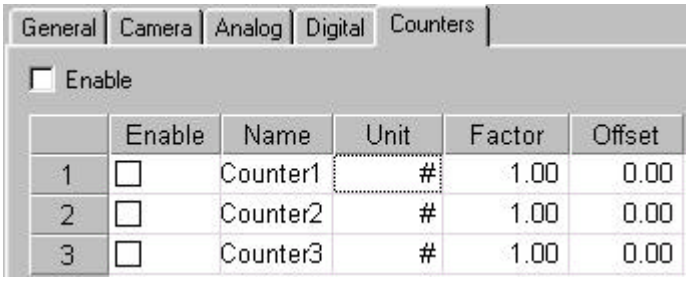



*Figure 11: measurement configuration - tab 'counters'*

**Enable**: enable/disable the individual channel

**Name**: keep the default name, or specify your own channel name by double-clicking and overtyping the name.

**Unit**: specify the unit you want the result to be expressed in
You can have the result immediately as a linear function of the number of increments:
Y = M * X + Q.

**Factor**: “Factor” is the “M”, the linear scale factor.

**Offset** “Offset” is the “Q” in the linear scaling

# 10. Measuring single LED's or objects

This chapter describes the actual measurement of LED's.
Should you encounter a problem, please refer to the troubleshooting chapter "Measuring single LED's or objects" on page 32.

If you finished the configuration is, click on "OK". The measurement form will pop up.

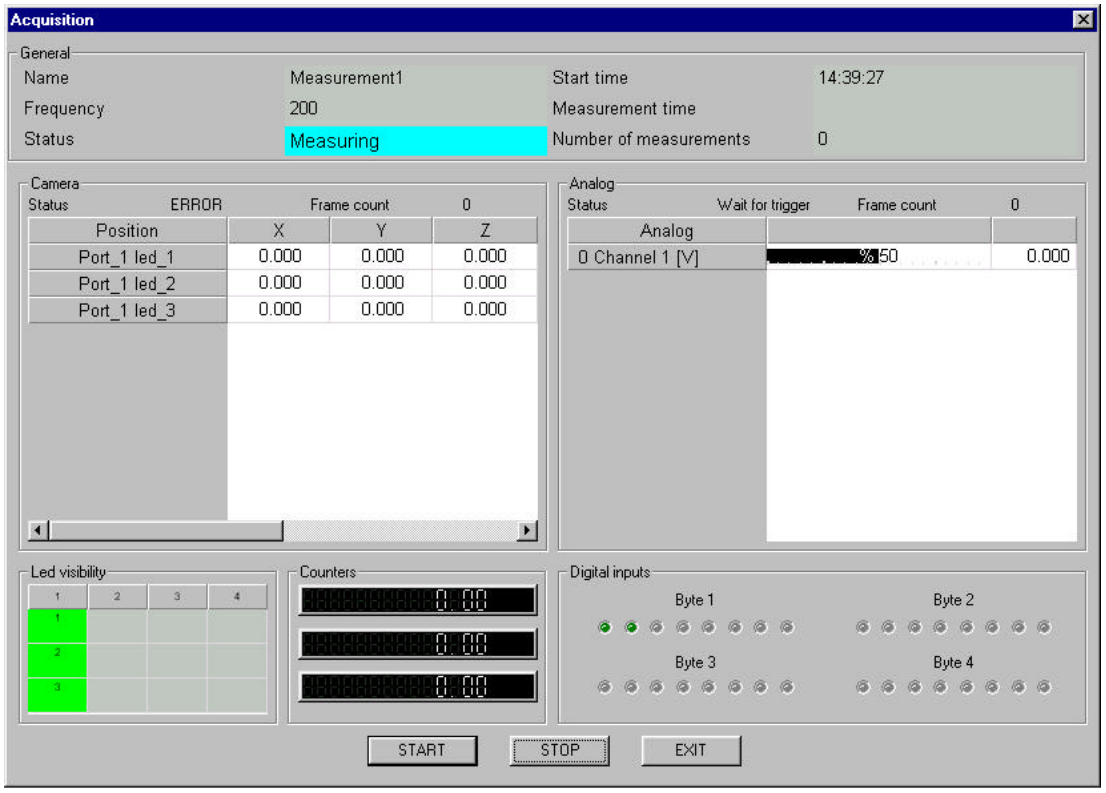



*Figure 12: acquisition – status window*

This form is divided into 6 information fields:

**General**: shows the name of the measurement, frequency, status of the measurement, and measurement timing.

**Camera**: during the measurement, this field will show for each LED the coordinates in camera frame. The display is not updated in real-time, the camera readings are sampled at certain intervals. It also displays the number of frames (snapshots) acquired by the camera.

**Led visibility**: during the measurement, this field will show whether each LED is visible or not. The display is not updated in real-time, the camera readings are sampled at certain intervals.

**Analog**: during the measurement, this field will show for each configured analog input channel its momentary value. The display is not updated in real-time, the ADC600 readings are sampled at certain intervals.

**Digital**: during the measurement, this field will show for each configured digital input its momentary state. The display is not updated in real-time, the ADC600 readings are sampled at certain intervals.

**Counters**: during the measurement, this field will show for each configured counter channel the momentary value. The display is not updated in real-time, the ADC600 readings are sampled at certain intervals.

If you selected external clocking, make sure the clock signal is connected to the ADC600 “ext CLK in” – input, or to the dedicated device.
If you selected “User” as start trigger, click with the mouse on “Start” : the measurement will start.
If you selected “User” as stop trigger, click with the mouse on “Stop” to halt the acquisition.

If you selected a world frame (see subchapter “Tab ‘camera’ “ on page 27), the software will start calculating each configured dynamic frame with respect to the chosen static world frame. The frame positions will be stored in files in a sub-folder “Position6D”. Individual LED’s will be calculated in this frame too, and will be stored in a sub-folder “Position3D”.

If you selected “Backup 3D camera data”, a backup will be made of the individual LED positions in camera frame. The backup data will be stored in a sub-folder “camera”

If you selected “Calculate residual deviations” (see subchapter “Tab ‘camera’ “ on page 27), the software will calculate a time history of the residual errors of frames.

# 11. Calculating result files

Sometimes, the automatic calculations after an acquisition are not sufficient, and you'd like to perform extra calculations: this can be the calculation of an extra frame, to express measurements in another frame, convert data units, …

Since DMM Modular allows for custom modules to be added, the number of applications is unlimited.Therefore, we restrict the possible postcalculations to the most common ones.

## 11.1. Calculation module 'Acquisition'

This standard module gives the user all the tools to measure, create 6D motion data files, and express data in different reference frames.

### 11.1.1. The function 'measure'

This function has been described in chapters "Configuring a measurement" on page 26 and "Measuring single LED's or objects" on page 32.

### 11.1.2. The function 'Create 6D data'

The function 'Create 6D data' is only useful for 6D-motions. It calculates the position and orientation of a dynamic frame of an object, based on the positions of the LED's linked to that object.

**Result**: Enter a name for the result file. This file will hold for each moment in time the position and orientation of the dynamic object frame with respect to the LED's frame of reference.

**Measurement**: browse to the file holding the raw LED data.

**Identification**: browse to the dynamic frame as identified for this object ( see chapter "Configuring DMM Modular" on page 24)

**First LED**: first led in the file <Measurement> that is part of the dynamic frame <Identification>. This value is filled in automatically, but can be changed anyway.

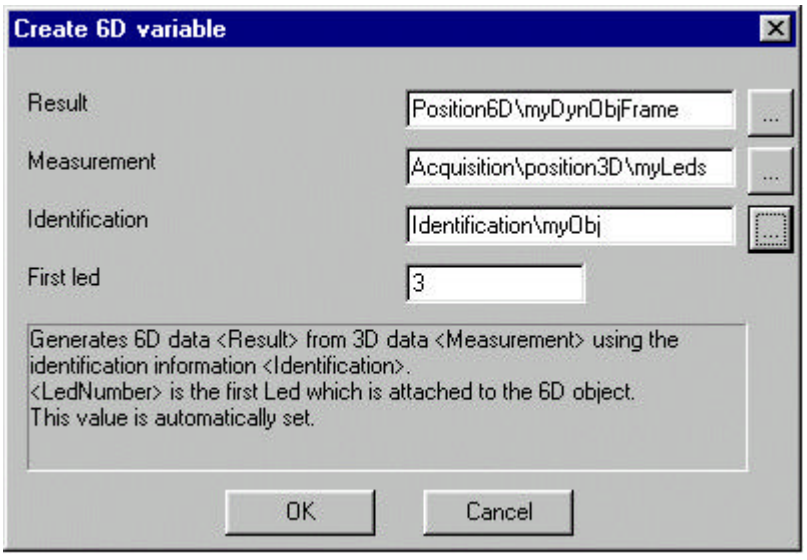



*Figure 13: 'create 6D variable' pop-up form*

### 11.1.3. The function ‘Transform’

The function ‘Transform’ recalculates 3D or 6D data files from one reference frame to another (e.g. from camera frame to a world frame).

If you want to transform the data into a static frame of reference (frame did not move with respect to camera), select a static frame in the ‘Identification’ folder.
If you want to transform the data into a dynamic frame of reference (frame did move with respect to camera), select a position6D data file in the ‘Position6D’ folder.

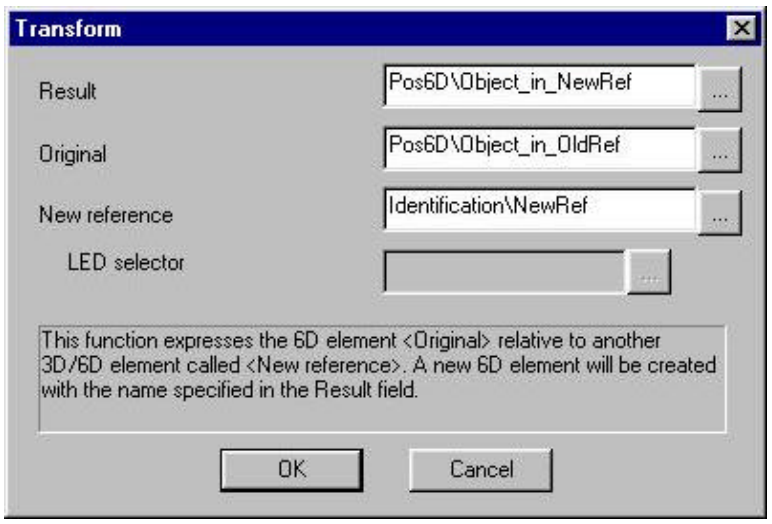



*Figure 14: 'Transform' pop-up form*

**Result**: Enter a name for the result file. This file will hold for each moment in time the position and orientation of the point or frame with respect to the frame <newRef>

**Original**: Browse to the data file with the position of the object <Object> expressed in the old reference frame <oldRef>

**New reference**: Browse to the new static or dynamic frame <newRef>

**LED selector**: applies only if a measurement file holding 3D-data is selected as new reference. If in this case a LED is selected, the object <Object> will be expressed in a reference frame that has axes, parallel to the axes of the frame <OldRef>, but whos origin lies at the position of the selected LED throughout the measurement.

## 11.2. Calculation module ‘Base’

This module provides the user with some standard, post-calculation tools.

**Note:** DMM Modular comes in two versions, ‘Standard’ and ‘Advanced’. The ‘Advanced’ version has more functions. The following sub-chapters will describe all of the functions – but for each function will be specified wether it is available in the ‘Standard’ or ‘Advanced’ version.

### 11.2.1. The function 'Offset'

This function offsets an entire data file. Purpose is to add/subtract a fixed offset in each direction, so that the start or the end of a measurement coincides with the origin of the reference frame, with the mean value of each direction, or the static position of another element

This function is available in the 'Standard' and 'Advanced' version.

**Warning! For 6D-frames, the function will also offset the angles : this will result in erroneous orientation angles.** The results yielded for 3D data files, 6D translations-only, or analog channels are correct.

If you want to offset the 6D orientations in a proper way, create with Geoloc a static frame for the requested position, teleport it to DMM Modular and transform the data file into this static frame.

**Output**: enter a data file in which for X,Y and Z (and the three angles) a fixed offset is added or subtracted.

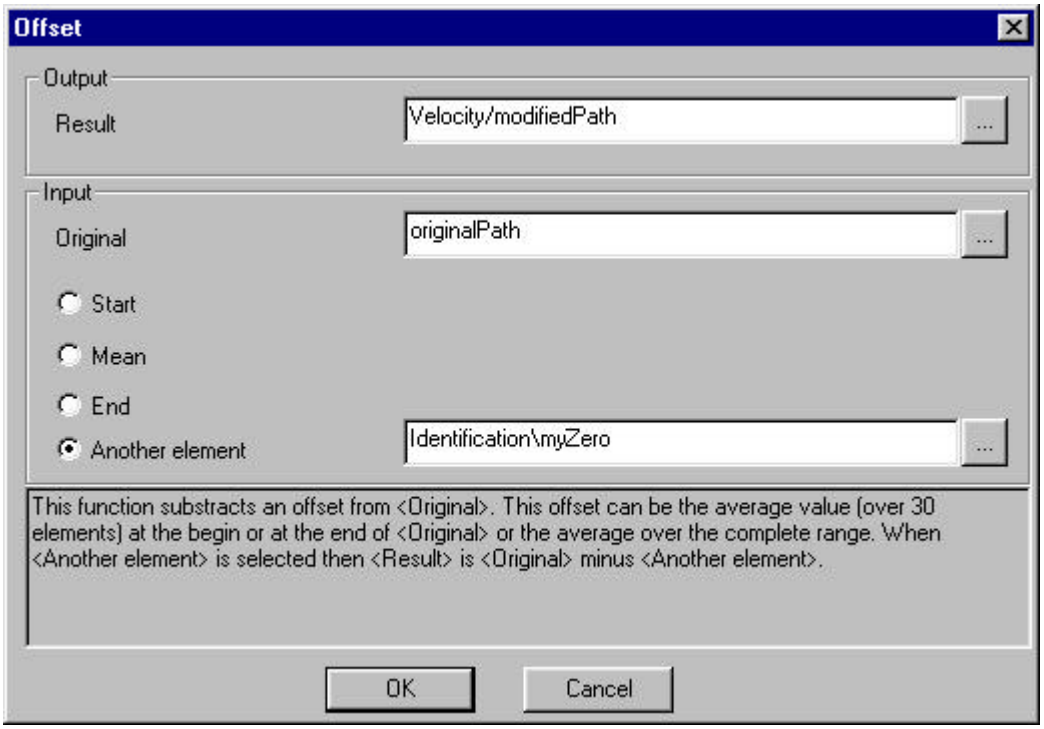



*Figure 15: 'Offset' pop-up form*

**Input**: browse to an object's data file "as-is". This can be a 3D-file, a 6D-file or a file with analog values. With the radio buttons, you can select what values you want to use as offset :

a. the start-value : <result> equals <original> minus the average of the 30 first positions. Ideally, this means that the very first position in the output file will have coordinates (0,0,0) ( (0,0,0,0,0,0) for a 6D-file)
b. the end-value : <result> equals <original> minus the average of the 30 last positions. Ideally, this means that the very last position in the output file will have coordinates (0,0,0) ( (0,0,0,0,0,0) for a 6D-file)
c. the mean value : this means that the average of all the positions will be 0.
d. Another element : choose a static element. The yielded coordinates will always be the distance between the object and this other element.

### 11.2.2. The function 'Distance'

This function calculates the 3D-distance between two elements.

This function is available in the 'Standard' and 'Advanced' version.

This function calculates the distance $|D| = \sqrt{(\Delta X^2 + \Delta Y^2 + \Delta Z^2)}$ between two element time series. An element can both be a 6D measurement file and a LED position file.

When choosing this function, the distance pop-up form appears:

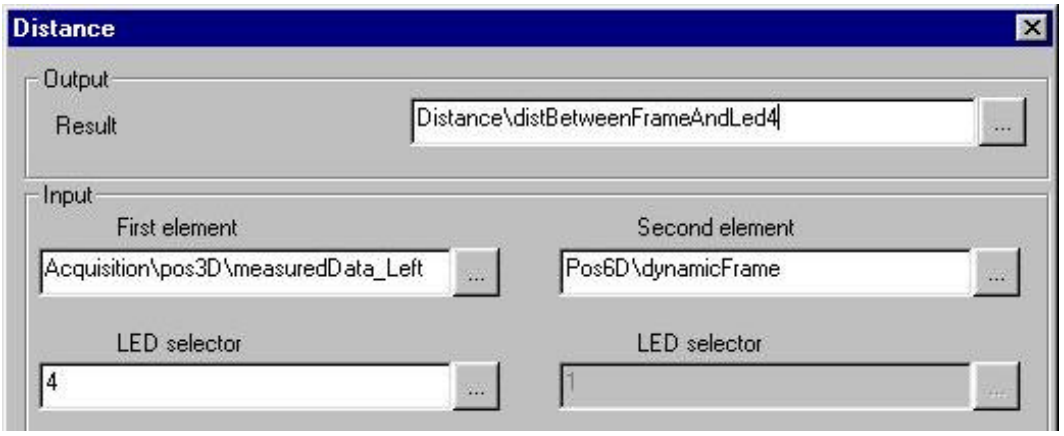



*Figure 16: the 'distance' pop-up form*

Parameters:

**Result**: name of the data file in which the distance data will be stored.

**First element**: name of the data file of the first of two elements between which to calculate the distance.
**LED selector**: if <First element> is a 3D-file, you must enter the LED number of the LED for which to calculate the distance to <Second element>. Clicking the Browse button (button with 3 dots) calls the following pop-up screen, from which you can select the LED name.

E.g. selecting the led "Wheel_1" and clicking "OK" will yield LED number 4.

**Second element**: name of the data file of the second of two elements between which to calculate the distance.
**LED selector**: if <Second element> is a 3D-file, you must enter the LED number of the LED for which to calculate the distance to <First element>. Clicking the Browse button (button with 3 dots) calls the pop-up screen, from which you can select the LED name.

Since a distance is an absolute value, it has no importance which of the two elements is chosen to be the first element, and which is chosen to be the second element.

### 11.2.3. The function 'Calculate velocity'

This function differentiates a position file by the time (first column), thus calculating the velocity of the point/frame in three/six dimensions :

This function is available in the ‘Standard’ and ‘Advanced’ version.

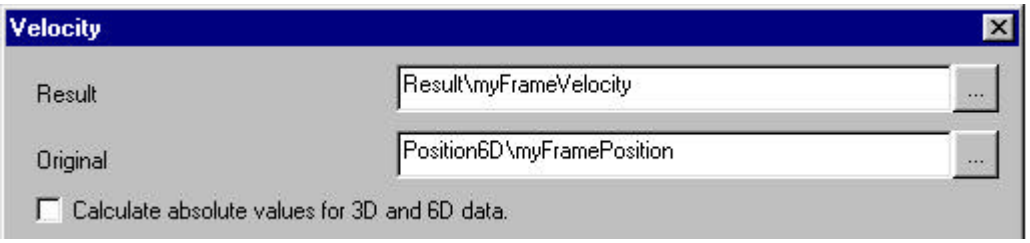



*Figure 17: 'Velocity' pop-up form*

**Result**: enter a name for the velocity file. The velocity is calculated for each axis independently (no 3D-velocity). For a frame, the angular speeds are calculated

**Original**: Browse to the position file that you want to be derived vs. time.

**Calculate absolute values …** : if checked, the function will calculate the “scalar” velocity, that is, to calculate the norm of the velocity vector (so all directional data is lost); if unchecked the velocity per degree of freedom is calculated.

### 11.2.4. The function ‘overshoot’

This function calculates the overshoot of a motion (the maximal amplitude excursion of a point with respect to its final position). This function is especially usefull for the automotive to analyse the results of doorslam testing.

This function is available in the ‘Standard’ and ‘Advanced’ version.

This function will find for the maximum overshoot of a point. This overshoot will be added to the result file, so subsequent tests can be joined into 1 data file and examinded together. At the same time, the function will create a copy of the original data, but offsetted to the final position of the point: this focusses later graphs optimally on the area of interest.

When choosing this function, the overshoot pop-up form appears:

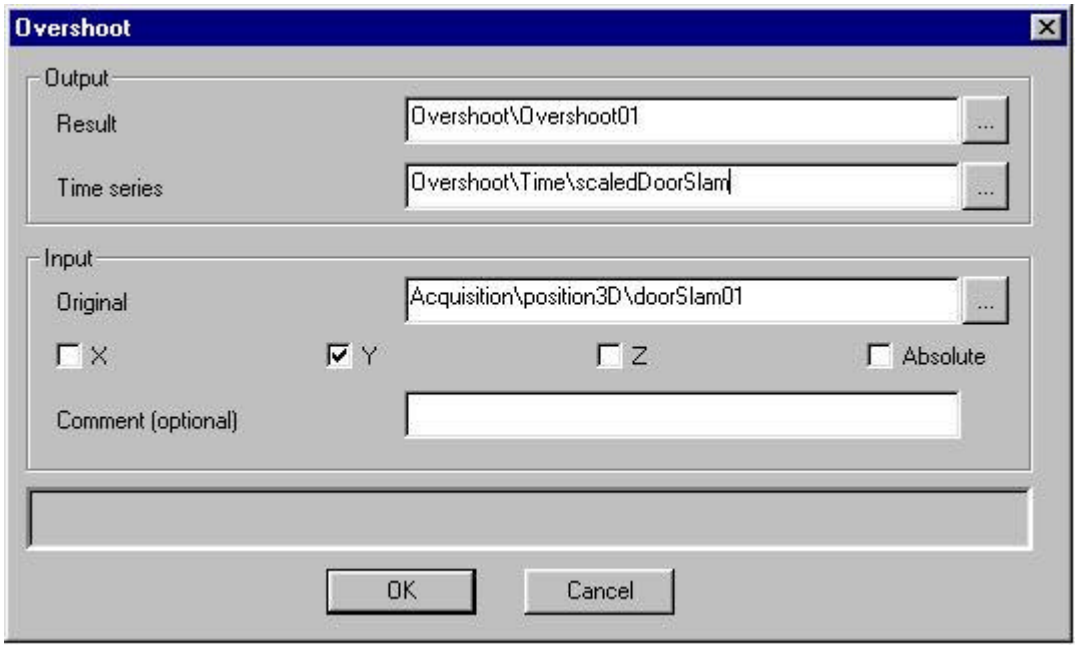



*Figure 18: the ‘overshoot’ pop-up form*

Parameters:
**Result**: name of the data file to which the overshoot will be added as a new line.

**Time series**: data file in which the original data is copied, but with an offset that places the final position of the motion in (0,0,0). This parameter is optional: leaving the text box blank will not calculate a time serie.

**Original**: the 3D- or 6D data file for which the excursion has to be calculated. For 6D-data files, the overshoot in X, Y and/or Z of the frame origin is calculated. For 3D-data files, the X, Y and/or Z of each led is calculated.

**"X"-, "Y"-, "Z"- and "absolute"-checkboxes**: if checked, the overshoot will be calculated for motions along this axis of the reference frame. If "absolute" is checked, the overshoot of the path in space is calculated.

**Comment:** optional comment field.

### 11.2.5. The function 'statistics'

The function "statistics" groups 4 statistical functions: find the maximum, find the minimum, calculate the average value and calculate the standard deviation.Each of these functions yield one or more numbers, which can be added to the result files. This way, results from multiple measurements can be compared.

These functions are available in the 'Standard' and 'Advanced' version.

#### *11.2.5.1 The function 'Find maximum'*

This function finds the maximum value for X, Y and Z (and the angles, if applicable) in a data file."Maximum" refers to the absolute value: negative values are taken into account too.

Parameters:
**Result**: name of the data file to which the maximum(s) will be added as a new line.
**Original**: name of the data file in which to find the maximum(s)

**Comment:** optional comment field.

#### *11.2.5.2 The function 'Find minimum'*

This function finds the minimum value for X, Y and Z (and the angles, if applicable) in a data file."Minimum" refers to the absolute value: negative values are taken into account too.

Parameters:
**Result**: name of the data file to which the minimum(s) will be added as a new line.

**Original**: name of the data file in which to find the minimum(s)

**Comment:** optional comment field.

#### *11.2.5.3 The function 'Calculate mean'*

This function calculates the mean value for X, Y and Z (and the angles, if applicable) in a data file.

Parameters:
**Result**: name of the data file to which the mean values will be added as a new line.

**Original**: name of the data file from which to calculate the mean values

**Comment:** optional comment field.

#### 11.2.5.4 *The function 'Calculate std'*

This function calculates the standard deviation for X, Y and Z (and the angles, if applicable) in a data file.

Parameters:

**Result**: name of the data file to which the standard deviations will be added as a new line.

**Original**: name of the data file from which to calculate the standard deviations

**Comment:** optional comment field.

### 11.2.6. The function 'export'

This function exports a data file into a *.CSV (comma-separated-value) ASCII-file, RPC3-format or a Matlab-compatible binary file ( *.MAT) .

This function is available in the 'Standard' and 'Advanced' version.

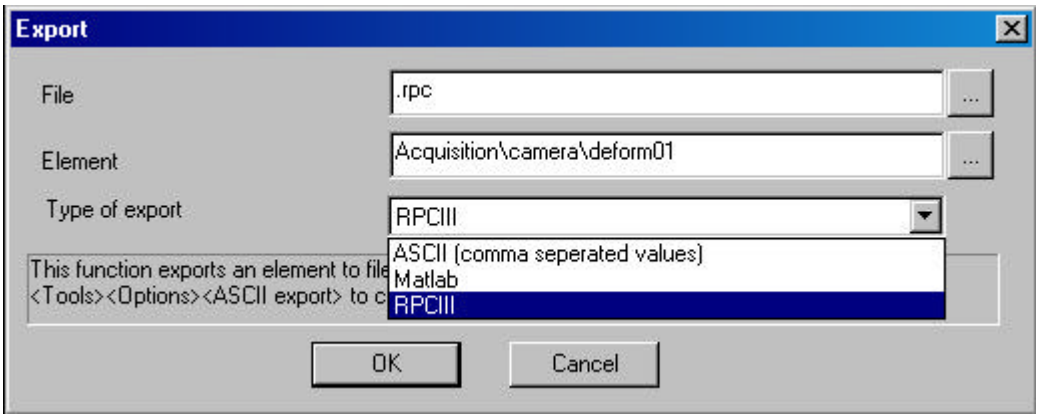



*Figure 19: 'file export' pop-up form*

Parameters :
**Type of export:** "RPC3" or "ASCII" for text files or "MATLAB" for binary, MATLAB-compatible files. Changing the file type will automatically change the default extension of the dump file.

**File:** name of the file in which to dump a measurement file. For ASCII files, the value separator and the default file extension are configured in the settings ( see chapter " 'Options' " on page 62).

**Element:** the data file to export.

### 11.2.7. The function 'Change orientation convention'

This function changes the orientation convention (see chapter "Orientation conventions" on page 19). The default orientation convention can be configured in the menu "Tools" – "Options" ( see chapter " 'Tools' menu" on page 62).
This function is only available in the 'Advanced' version.

Change orientation convention
Result
Position6D\myOtherAngularConvention
Original
Position6D\myPositionFile
New orientation convention
Roll Pitch Yaw
This function changes the orientation convention of <Original> and stores the result in <Result>.The new orientation convention can be ROLL PITCH YAW (ZYX rotation around fix axis), STEER CAMBER SPIN (ZXY euler angles) or SKREW vector (equivalent rotation)
OK
Cancel

*Figure 20: 'change orientation convention' pop-up form*

**Result**: enter the name for the resulting data file.
**Original**: enter a name for the data file whose orientation convention must be changed.

**New orientation convention**: choose the new orientation convention in which the data file will be expressed.

### 11.2.8. The function 'Calculate point in 6D frame'

This function calculates the position of a 3D-point in a chosen reference frame, based on the coordinates of this point with respect to a dynamic object frame, and the 6D-position of that dynamic frame in the reference frame. E.g. we know the coordinate of the corner of a cube in the cube object frame; we know how the cube frame (with its origin in the middle of the cube) moved with respect to a world frame; so we can calculate how the corner point moved with respect to the world frame.

This function is available in the 'Standard' and 'Advanced' version.

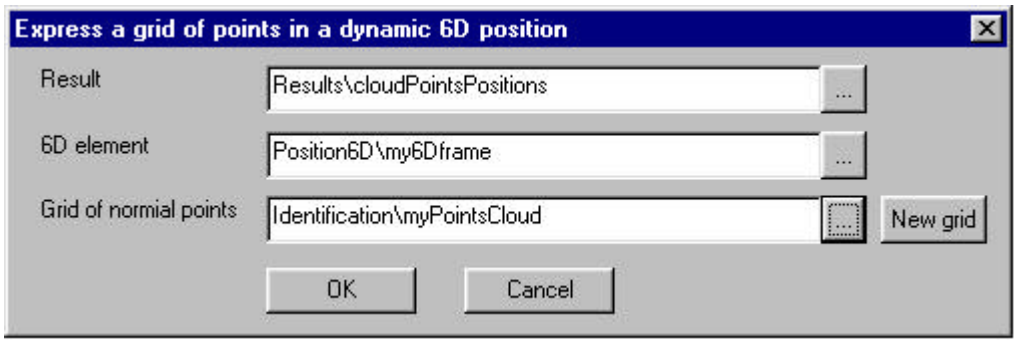



*Figure 21: 'calculate point in 6D frame' pop-up form*

**Grid of nominal points**: this is the point cloud, a set of multiple points of which you want to calculate the positions. Either this point cloud was teleported from GeoLoc, or you can create a new cloud on-the-spot by clicking "New Grid" : enter the number of points in the cloud, click on "set", and enter the coordinates in the grid.

**6D element**: Browse to the data file of the dynamic frame, in which the point cloud is expressed. The dynamic frame is expressed in a reference frame

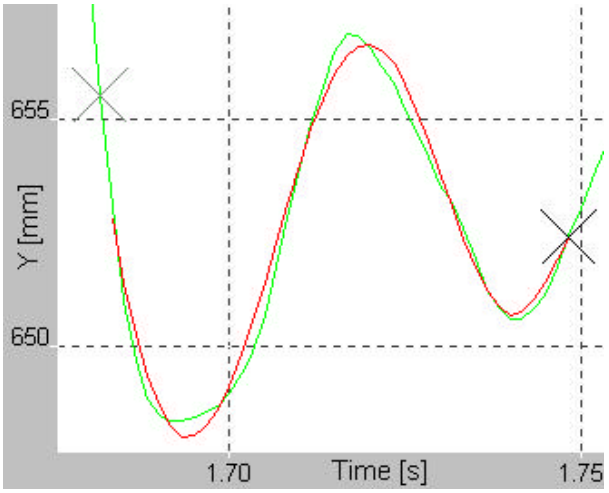
**Result**: Enter the name of the data file in which for each point of the cloud the position is calculated with respect to the same reference frame in which the dynamic frame is expressed.

### 11.2.9. The function 'fit polynome'

This function fits a polynomial function through two sets of data points. Once this is established, mathematical analysis becomes much easier.

This function is available in the 'Standard' and 'Advanced' version.

Let's take for example a car door that closes. Once the door hits the rubber seals, a small vibration occurs. It would be convenient if the lateral position (Y-motion) were a polynomial, because the speed and decceleration would be very easy to calculate (1st and 2nd derivation vs. time).
Therefore, we'll fit a polynomial function through the raw measured data.




This is the result: the green graph is the raw data; the red graph is the polynomial fit of Y vs. time between T=1.682 seconds and T=1.75 seconds.
The two sets of data in this case are the lateral motion, and time.
We express them in general as $Y = ? a_n(X)^n$ , where *X* is the independent variable and *Y* is the dependant variable ( in this case, *X* is Time, and *Y* is the Y-motion of the door).

The function permits to select only a fraction of the data set, so the polynomial fit will not be disturbed by transient or DC effects outside the zone of interest. Ranges can be set for both for *X* and *Y* where the fit should be limited to.

The result of the function is a new data file with three columns :
- ??*X* ( the independent variable)
- ??*Y'* ( the calculated, polynomial dependant variable)
- ??*Ey* ( the error between each *Y*–value and his polynomial *Y'*–value )

When choosing this function, the polynomial-fit pop-up form appears.

Parameters:
**Name**: name of the calculated data file.

**X - data**: data file from which to retrieve the independent variable ( *X* ).

**X - Channel**: once the file is selected, you can select the variable that will be the independent variable of the polynomial fit.

**X - from**: if unchecked, the range lower limit will be the first value of the X-channel. If checked, the user must specify the range lower limit.

**X – to:** if unchecked, the range upper limit will be the last value of the X-channel. If checked, the user must specify the range upper limit.

**Y - data**: data file from which to retrieve the dependent variable ( *Y* ).

**Y - Channel**: once the file is selected, you can select the variable that will be the dependent variable of the polynomial fit.

**Y - from**: if unchecked, the range lower limit will be the first value of the Y-channel. If checked, the user must specify the range lower limit.

**Y – to:** if unchecked, the range upper limit will be the last value of the Y-channel. If checked, the user must specify the range upper limit.

**Degree of fit**: highest power of *n* in the fit. Remember you need at least *n*+1 data points to make a fit ( 2 points for a linear fit, 3 points for a quadratic fit, 4 points for a cubic fit, …).

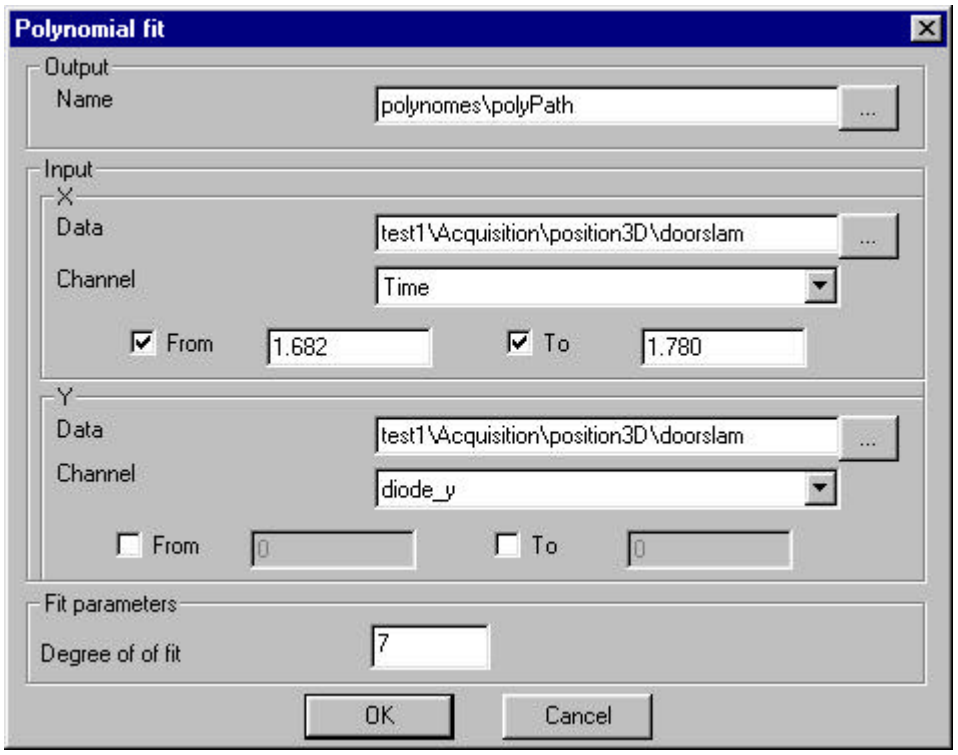



*Figure 22: the 'polynomial fit' pop-up form*

The polynomial coefficients can be obtained from the file properties ( see chapter "The workspace window" on page 23 ) :

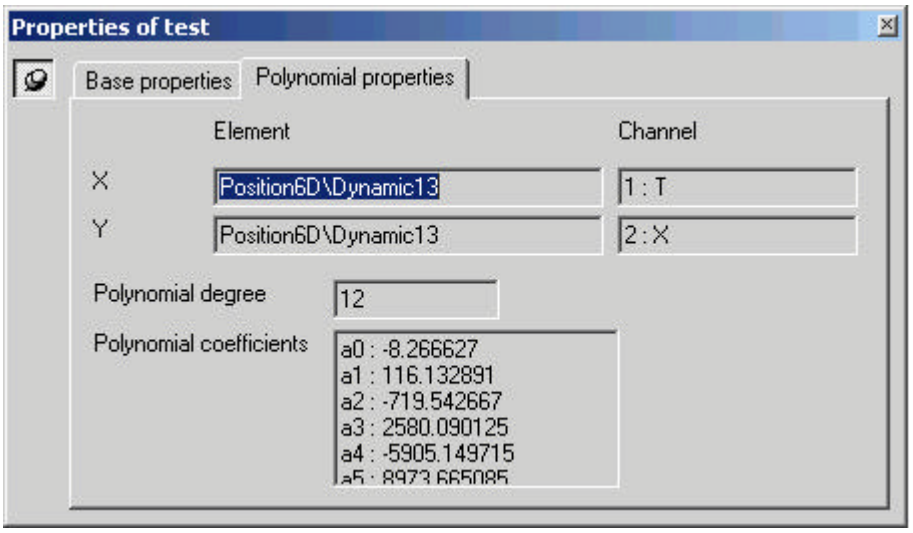



*Figure 23: polynomial fit - the polynomial coefficients*

### 11.2.10. The function 'Filter'

This function applies a chosen digital filter to measured or calculated data. This function can be used to filter noise from a signal ("low-pass" filter), or to remove DC-components from a signal ("high-pass" filter).

This function is only available in the 'Advanced' version.

This function applies an algorithm to a data file that corresponds to a classic Butterworth low-pass or high-pass filter. This digital filter does not shift the filtered data with respect to the unfiltered data (as do analog filters). Digital filters do NOT resolve alias-effects!

When choosing this function, the filter pop-up form appears:

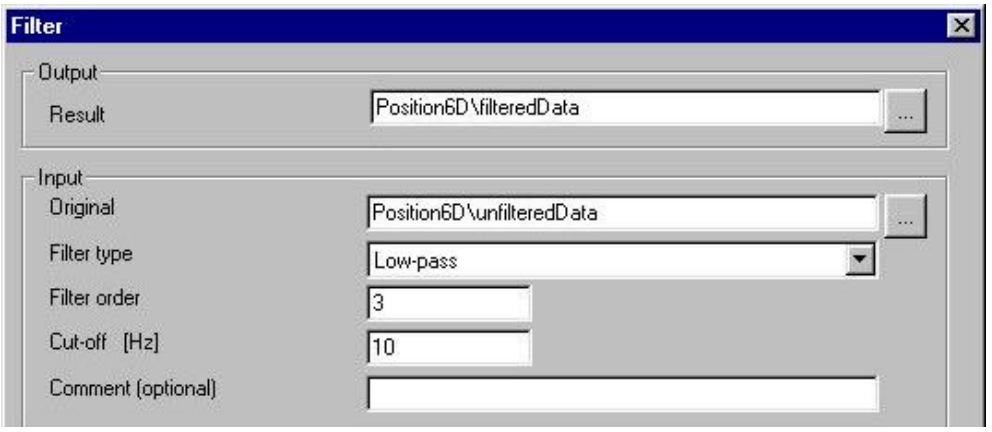



*Figure 24: the 'filter' pop-up form*

Parameters:

**Result**: name of the data file in which the filtered data will be stored.

**Original**: name of the data file which holds the data to filter.

**Filter type:** select "low pass" to filter out all high frequency components, select "high-pass" to filter out all low frequency components.

**Filter order:** specify the filter order. For every order, an attenuation of 20 dB per frequency decade is obtained.

**Cut-off [Hz]:** specify the filter cut-off frequency (the frequency which is attenuated by 3 dB). All higher/lower frequencies (for a low-pass/high-pass filter) will be attenuated stronger, depending on the selected filter order.

**Comment:** optional comment field.

### 11.2.11. The function 'Frequency'

This function performs the spectral analysis of a given signal.

This function is only available in the 'Advanced' version.

This function calculates the frequency spectrum of each variable in a given data file, that is, try to obtain the same time series by adding an infinite number of sines with different frequencies and amplitudes. Due to mathematical limitations, and the fact that we are

sampling signals in a discrete manner, the highest frequency that can be calculated is half the measurement frequency ( E.g. we measure at 100 Hz, so the highest frequency in the calculated spectrum will be 50 Hz)

When choosing this function, the frequency pop-up form appears:

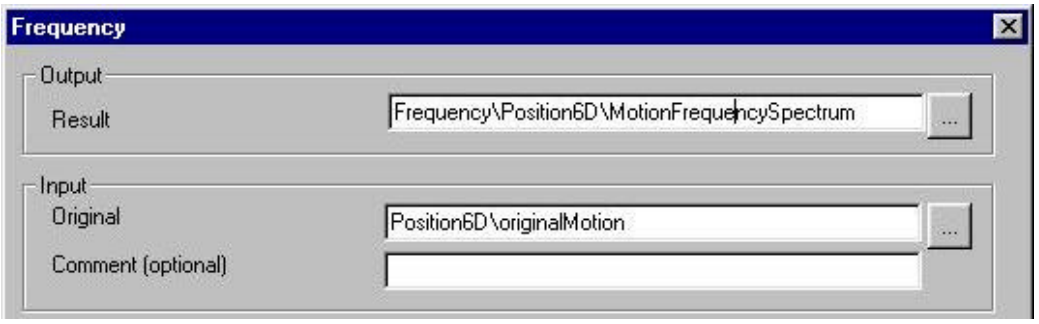



*Figure 25: the 'frequency' pop-up form*

Parameters:

**Result**: name of the data file in which the frequency spectrum will be stored.

**Original**: name of the data file from which a spectral analysis is required.

**Comment:** optional comment field.

### 11.2.12. The function 'Join measurements'

This function applies only for situations where synchronized data from two PC's must be joined on one PC to perform calculation. A good example is the measurement of two car wheels, for which the relative angles are to be calculated on 1 PC.

This function is only available in the 'Advanced' version.

Before executing this function, make sure the two data files are present in the project workspace; the data file from the other PC must be imported, either by the menu option "Add From file" (see chapter "'Project' menu" on page 60 ), or by the script function addfromfile().

It is the responsability of the user that the two data files are expressed in a common reference frame. If the names of these reference frames are different, a warning will be generated, but the function will be executed anyway.

Since the join-operation happens on measured data level, most often it will also be necessary to import the corresponding identifications from the second PC.

When choosing this function, the 'join measurements' pop-up form appears:

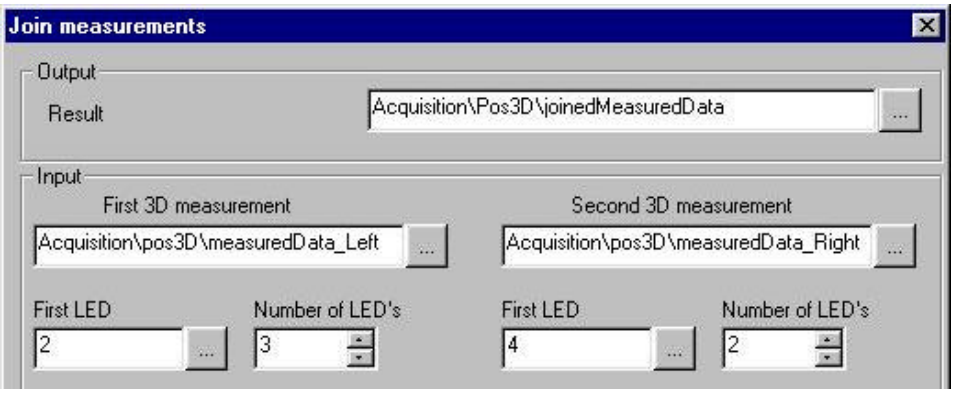



*Figure 26: the 'join measurements' pop-up form*

Parameters:

**Result**: name of the data file that was joined from the two input data files.

**First 3D measurement**: name of the data file who will be put first in the joined data file.
**First LED:** you can extract a limited set of led's from a measurement file. Enter here the index (starting from 1) of the first led to be part of the joined file …
**Number of LED's** … and enter here the number of leds, including <First LED>, to be copied into the result file

**Second 3D measurement**: name of the data file who will be put last in the joined data file.
**First LED:** you can extract a limited set of led's from a measurement file. Enter here the index (starting from 1) of the first led to be part of the joined file …
**Number of LED's** … and enter here the number of leds, including <First LED>, to be copied into the result file

For example, the screenshot above will produce the following joined measurement file :

Led 1 = led 2 from Acquisition\Pos3D\measuredData_Left (first led)
Led 2 = led 3 from Acquisition\Pos3D\measuredData_Left
Led 3 = led 4 from Acquisition\Pos3D\measuredData_Left
Led 4 = led 4 from Acquisition\Pos3D\measuredData_Right (first led)
Led 5 = led 5 from Acquisition\Pos3D\measuredData_Right

## 11.3. Adding Calculation modules

It is possible to let us convert your own written Matlab routines into a module, in which case you'll have a better integration with DMM Modular. This binary module will be sent to you; all you have to do is to copy the module in the <*root*>\bin-directory of Rodymdmm2 ( by default, this is *c:**rodymdmm2*\bin ), and restart DMM Modular.

If you do have code, that you want to integrate, or you have additional features you want to calculate, send an E-mail with your request to support@krypton-intl.com .
To get more details, or information on the formal specifications, send an E-mail to the same address.

In case you can't wait for the compiled module, or you want to "improvise" when manipulating data, please refer to the chapter "Exporting to Matlab" on page 59.

In case you encounter problems with installing new modules, please refer to the troubleshooting chapter "Adding calculation modules" on page 46.

# 12. Creating spreadsheets & graph's

After all the calculations have been performed, you can visualize your results.
Should you encounter a problem, please refer to the troubleshooting chapter "Creating spreadsheets & graph's" on page 47.

DMM Modular offers three data representations :

1) Spreadsheet : displays the data in a two-dimensional matrix. Use this form when you are interested in the numerical values
2) 2D-graphs : plots two variables against each other. Use this mode to see trends in certain directions. It's fairly easy to see values too.
3) 3D-view : shows a 3D-view of the objects in camera frame. Use this mode to see what really happened in slow-motion. It also allows a replay-function, to see the movement of the objects in a fluent motion.

It is possible to see the same data in multiple views at the same time. The data in the different views is synchronized with **cursors**: when you add a cursor, every view positions itself to that moment in time. Since cursor operations are common to all three, we'll start with the cursors.

## 12.1. Working with cursors

A cursor indicates 1 sample at a time. It can be used as index when investigating values, or when looking at spreadsheets, 2D-graphs or 3D-views.

You can define multiple cursors.
A special kind of cursor is the difference cursor: this cursor makes the difference between the values at which two other cursors are pointing. This is an easy feature to visually find the distance between a graph maximum and minimum, …

### 12.1.1. Working with single cursors

All cursor values are displayed in the output window. Select the tab "Cursors" in the output window:

Move the mouse into the white area of the output window and click right; select "Cursor", then select "Add".

In the white window, a line will appear, labeled "C*x*" where *x* is the cursor number.

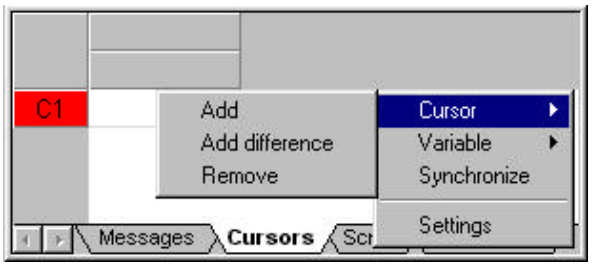



*Figure 27: cursor tab in output window*

The cursor line will have no columns yet, because there are no variables selected yet.

Move the mouse into the white area of the output window and click right ; select "Variable", "Add" : a pop-up labeled "Variable selection" will appear with all available files. Select one file and click OK.

Next, a listbox labeled "Select channels" will appear. Select all the variables you want to track from this listbox, and click "OK".

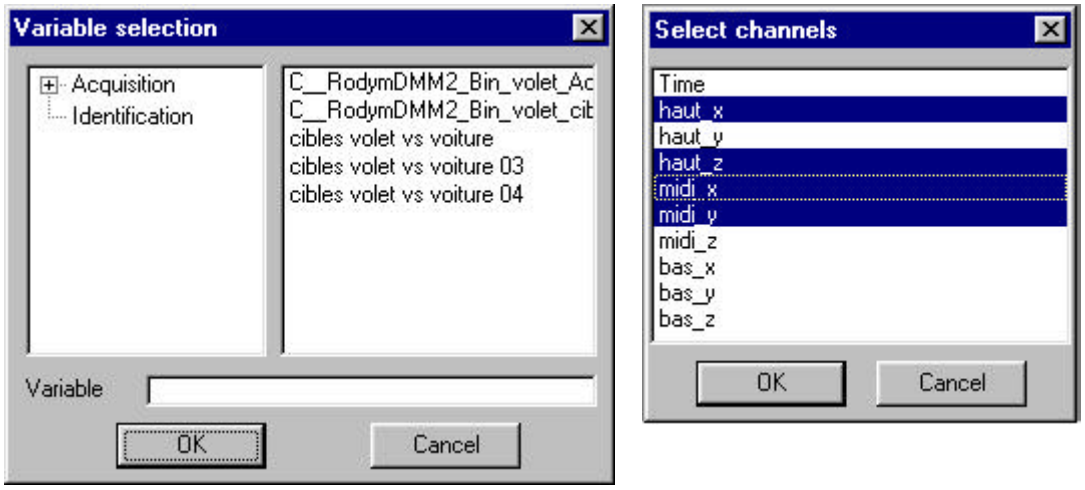



*Figure 28: cursor value selector forms*

Every selected variable is assigned to a column of the cursor. The upper column heading states the foldere name from which the variable is read. The lower heading states the name and unit of the variable:

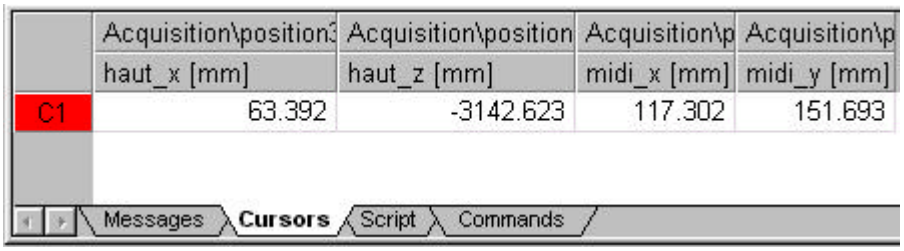



*Figure 29: cursor column headings*

To remove a variable, right-click in the white area of the output window, and choose the option “Variable” – “Remove”. A list-box will appear with all the existing variables: select the variable(s) you want to remove and click “OK”.

To navigate a cursor through the samples, use the “Arrow left” or “Arrow right” key on the keyboard; the cursor will move sample by sample through the data file.
Holding down the “Ctrl”-key during this navigation will make the cursor traversing the data file in “coarse” mode (skipping samples to increase the speed).

If you add additional cursors, they will use automatically all the already existing variables.

When working with multiple cursors, only one cursor moves at the same time: this is the “active cursor”. The active cursor is marked in red.
The newest cursor will automatically be the active cursor. To change the active cursor, use the “Arrow-up” or “Arrow down” key on the keyboard.

To remove a cursor, right-click in the white area of the output window and choose the option “Cursor” – “Remove”. A list-box will appear with all the existing cursors: select the cursor(s) you want to remove and click “OK”.

### 12.1.2. Working with difference cursors

As soon as two cursors exist, you can define a difference cursor : this cursor indicates the difference per variable between two cursors.
For example, create a difference cursor. Make cursor 2 active and move it to a minimum position of a certain motion. Now make cursor 1 the active cursor, and place cursor 1 at a maximum position: the difference cursor now indicates the maximum swing between those two positions.

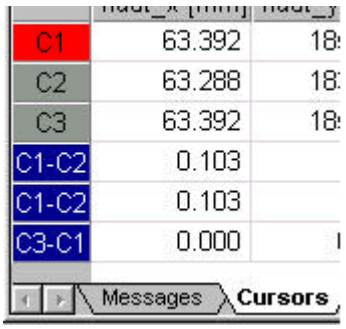
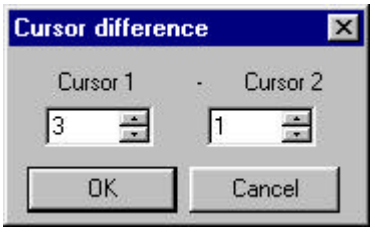



*Figure 30: difference cursors*

A difference cursor is always labeled "C*x* – C*y*" where *x* and *y* are the cursor numbers. Difference cursors are always marked blue.

When there are only two cursors, the only possible difference cursor is "C1 – C2".

As soon as you have three cursors, the following pop-up appears. Configure the two cursors you want to subtract. Select the cursor numbers *x* and *y*:




To remove a difference cursor, follow the same procedure as removing a single cursor ( see chapter "Working with single cursors" on page 47 ).

### 12.1.3. Remaining cursor options

If you right-click in the white area of the cursor tab, there are two options left: "Synchronize" and "Settings".

**Synchronize:** if you click synchronize, all cursors, both in the cursors tab and in the graph's and spreadsheets, are synchronised, i.e. each cursor (C1-Cx) is set at the same time sample for the different variables.

**Settings:** you can configure the number of samples to skip in "coarse" mode and the cursor column width (in pixels)

## 12.2. The spreadsheet view

The spreadsheet view displays the data in a two-dimensional view. The data can be searched for on a specific criterion, or the data can be exported as a comma-separated value, so it can easily be imported in Microsoft Excel.

To open a spreadsheet view, do the following:

choose from the menu the option "*Window*" – "*Open*" - "*spreadsheet*"
or, open the tab "Views" of the workspace window and double-click on "Spreadsheet"

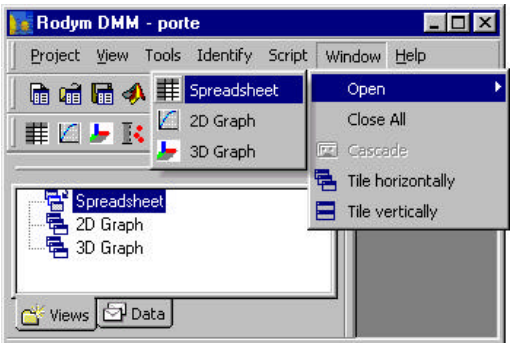
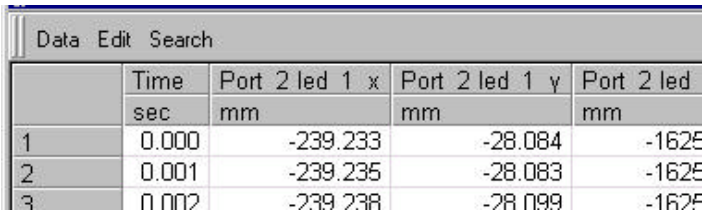



*Figure 31: opening a spreadsheet view*

or, click on the spreadsheet icon on the icon bar :

next, open the tab "Data" of the workspace window
drag the requested data file from the workspace window onto the spreadsheet, and drop it there.

The data will appear in a spreadsheet window. Each column corresponds to a measured value ( time, positions, rotations, … ). Above each column, the name of the variable and its units are listed.
Every line corresponds to a sample in time. On the left, the sample numbers are listed.




| | Time | Port 2 led 1 x | Port 2 led 1 y | Port 2 led |
|---|---|---|---|---|
| | sec | mm | mm | mm |
| 1 | 0.000 | -239.233 | -28.084 | -1625 |
| 2 | 0.001 | -239.235 | -28.083 | -1625 |
| 3 | 0.002 | -239.238 | -28.099 | -1625 |

*Figure 32: the spreadsheet view*

It is possible to edit the data in the file by double-clicking on a cell and editing the contents. **Attention ! The modified data permanently overwrites the old cell data in the data file !**

If you defined one or more cursors, ( see chapter Working with cursors on page 47 ), the sample number will get a white background and an annotation of the cursor that is currently there.

| | Time |
|---|---|
| | sec |
| 1 C1 | 0.00 |
| 2 | 0.00 |
| 3 C2 | 0.00 |
| 4 | 0.01 |

*Figure 33: cursor annotations*

The last colum is always empty. The user can enter text remarks in this column: these texts will appear as markers in 2D-graphs.

### 12.2.1. The 'Data' menu

The data menu allows you to trace dates, file creators, or to export the entire file in an ASCII text file.

**Properties:** to open the properties sheet of the frame whose data file we're looking at. See chapter "The workspace window" on page 23.
**Export**: exports the current data to *.CSV-file or *.MAT-file. See chapter "The function 'export'" on page **Error! Bookmark not defined.**.
**Close**: to close the spreadsheet

### 12.2.2. The 'Edit' menu

The edit menu allows you to modify the cell data using existing data in another spreadsheet.

**Copy:** to copy the spreadsheet partially or in whole to the clipboard. This data can be pasted "as-is" in Microsoft Excel. The menu offers three possibilities:

a. To copy/paste the entire spreadsheet, choose: **All** ( if the spreadsheet exceeds 1500 samples, the copy might start taking very long )
b. To copy/paste only the part you selected, choose: **Selection**
c. To copy/paste the results of a search query, choose: **Results of search**: (see chapter "The 'Search' menu" on page 51)

**Paste:** to copy a selection of spreadsheet cells into the opened spreadsheet, starting at the currently active cell. This data can be copied from Microsoft Excel. **Attention! The pasted data permanently overwrites the old cells in the data file!**

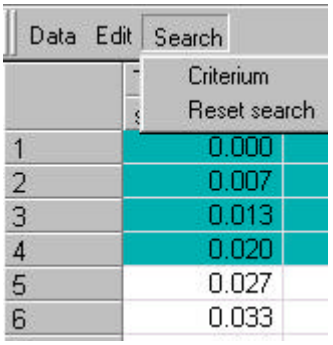
**Delete:** to delete selection of spreadsheet rows. If you have selected one or more rows, "delete" will remove these rows permanently from the data file. Before deleting the rows, a pop-up form appears to confirm or cancel the operation. **Attention! The operation is permanent and removed data can not be restored.** If you have selected a cell instead of a row, the row of that cell will be the target to be deleted.

### 12.2.3. The 'Search' menu

To search the spreadsheet for certain criteria, specify a search criterion, which looks for 1 sample, or for a set of samples.

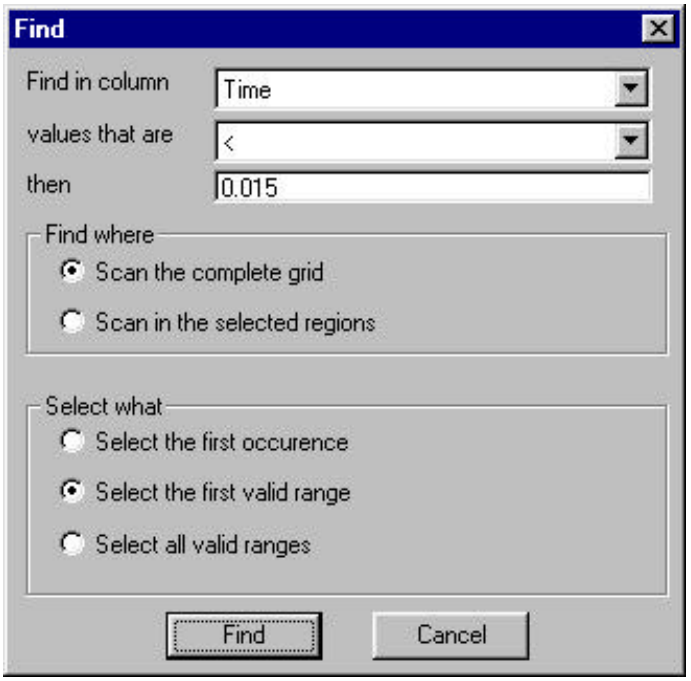
In case of a set of samples, all the samples that fit the criterion are selected in a **range**. A range is displayed with an ocean-green background.

Specifying that the query should only run in a previously selected range enables you to combine multiple search criteria.




#### *12.2.3.1 Criterium*

'Criterium' sets the search criterion for the next search:




Parameters:

**Find in column:** select the column to which the search criterion will be applied
**Values that are:** select the logical operator (<, >, <=, >=, =, <>, max, min) to be applied
**Then:** enter the numerical threshold value
**Find where:** select either the entire grid (entire spreadsheet), or select the "selected regions". The selected regions are those samples that fitted a previous search. The latter option allows you to combine search criteria.
**Select what:** specify whether you want to look for the first (single) sample that fits the criterion, the first range that fits the criterion, or all the ranges that fit the criterion.

#### *12.2.3.2 Reset search*

'Reset search' neutralizes the currently selected ranges and allows a fresh, new value.

## 12.3. The 2D-graph

The 2D-view plots a Y vs. X-graph of two variables. You can zoom, pan or scale the graphs. You can express values with respect to time, or to each other. You can add a cursor that runs over the graph, indicating the exact numeric value of a variable at a certain point on the graph.

To open a 2D- graph do the following:
choose from the menu the option "*Window*" – "*Open*" - "*2D-graph*"
or, open the tab "Views" of the workspace window and double-click on "2D-graph"

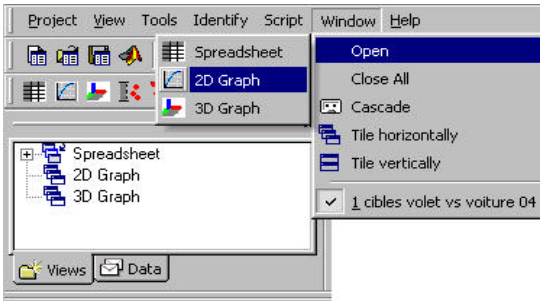



*Figure 34: opening a 2D-graph*

or, click on the spreadsheet icon on the icon bar :

next, open the tab “Data” of the workspace window
drag the requested data file from the workspace window onto the graph, and drop it there.

As soon as the file is dropped, a dual list box pops up. In the left list box, select the variable that will be plotted on the X-axis of the graph. You can select only one variable. In the right list box, select the variable(s) that will be plotted on the Y-axis. These can be multiple variables (shift-click to select a range, Ctrl-click to select a set of separate variables). Click on “OK” to confirm the choice:

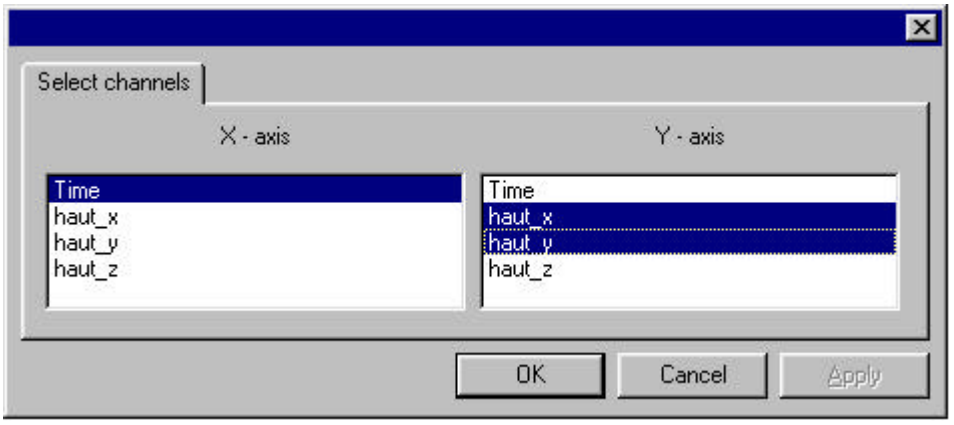



*Figure 35: 2D-graph variables selection pop-up form*

The 2D-graph will be opened. Initially, the graph is stretched to its maximum contents.

You can pan the graph by moving the mouse over the graph, holding down the left mouse key, and moving the mouse: as long as the left button is down, the graph will follow the movements of the mouse.

Scrolling the mouse wheel will zoom in and out on a graph.

If you hold down the key Ctrl, and then click the left mouse button, a rule will be displayed between the cursor position at the time of the left mouse click, and the current cursor position. The rule will display the difference in X and Y (units according to the units, specified on the X- and Y-axis), and the 2D-distance. If you release the Ctrl-key, the dX-, dY- and 2D-value will be shown in the message tab of the output window.

If you hold down the keys Ctrl + Shift, and then click the left mouse button, and you move the mouse from left to right, you can zoom the X-base of the graph in and out.

If you hold down the keys Ctrl + Shift, and then click the left mouse button, and you move the mouse up and down, you can zoom the Y-base of the graph in and out.

Right-click on the graph to reach the other options of a 2D-graph, a menu with the following options will appear:

**Zoom:** to draw a rectangle over the graph: the area inside the rectangle will be zoomed over the full extent of the 2D-graph

**Zoom user:** to bring up a pop-up that allows you to enter the extent of the zoom window numerically

**Zoom all:** to reset the graph in its original state

**Save as metafile:** to export the graph as a Windows meta-file (*.EMF). This allows you to import graphs as vector files in CorelDraw©

**Print:** sends the graph to the default printer

**Settings:** to allow you to show/hide the grid lines, and to change the font for the X and Y-labels. There is also a checkbox “Show markers” to display comment labels on screen as entered in the spreadsheet view.

**Show legend:** shows a legend form for this graph. The legend tells for each line plot what the line colour is, which variable is plotted in X and which variable in Y:

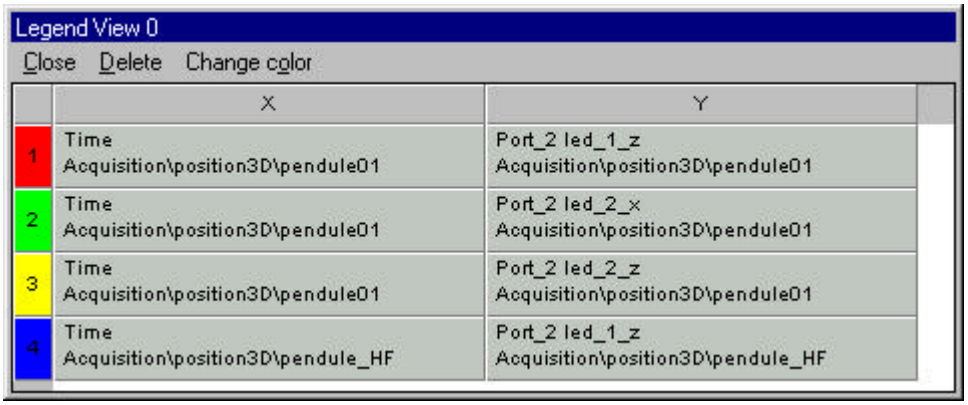
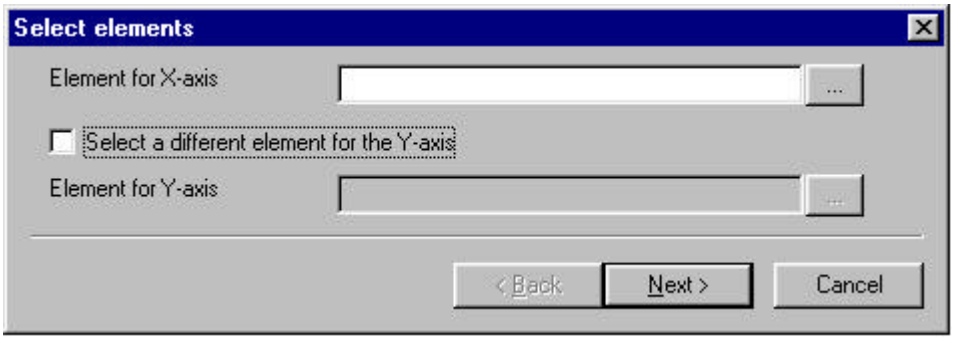



*Figure 36: 2D-graph legend window*

To delete a line plot, click on the X or the Y of the plot and choose “Delete” from the legend’s menu.
To change a plot colour, click on the X or the Y of the plot and choose “Change colour” from the legend’s menu. Select the colour you want, and confirm.
To close the legend, choose “Close” from the legend menu

**Add :** To add a line plot to the 2D-graph. **This option is interesting if you want to compare synchronised data from two files, e.g. an analog displacement sensor vs. the optically measured displacement.** The first pop-up asks for an element for the X-axis. Enter the name of the data file that describes the motion of an element.
If the variables for the Y-axis come from the same data file, uncheck “Select a different element for the Y-axis”. In case of a different data file, check the checkbox and enter the name of the other data file




The next pop-up is the dual list box as seen in the beginning of this chapter. Complete as previously described in the beginning of this chapter.

**Add cursor:** adds a cursor to the graphs, a cross that can navigate over the plots. **Don't forget to synchronize your cursors!** (see subchapter **Error! Reference source not found.** ).

## 12.4. The 3D-view

The 3D-view visualizes in slow-motion (or step-by-step) in a 3D-perspective the motion of a set of selected objects. The user can change the viewpoint, determine the start- and stop of the animation, and configure some of the "look and feel" settings.

To open a 3D- view do the following :
a. choose from the menu the option *"Window"* – "*Open*" - "*3D-graph*"
b. or, open the tab "Views" of the workspace window and double-click on "3D-graph"

c. or, click on the spreadsheet icon on the icon bar :



next, open the tab "Data" of the workspace window

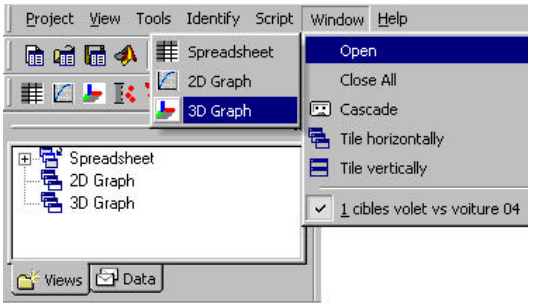


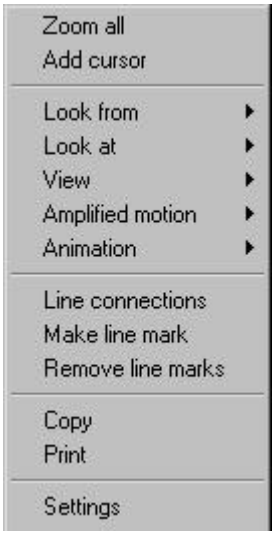

*Figure 37: opening a spreadsheet view*

You'll see a tri-coloured frame on a checkered field appear in the result window. This frame will be the reference frame. As soon as you drop a data file on the view, the reference frame in which the data file is expressed, will coincide with this frame. The colour convention equals "RGB" :
The X-axis is **RED**, the Y-axis is **GREEN**, and the Z-axis is **BLUE**

Next, open the tab "Data" of the workspace window. Drag the requested data file from the workspace window onto the 3D-view, and drop it there. **Do NOT drag dynamic frames or static frames into the 3D-view !** The information comprised in these files is differently interpreted by the animation: relying on their information might set you off on the wrong foot.

If the data file contains 3D-data, you'll see red spheres appear in the 3D-view, representing the LED's.
If the data file contains 6D-data, you'll see a frame appear in the 3D-view, representing the object frame. If the properties were set appropriately, you'll also see a rendered CAD model around the object frame.

Click right in the 3D-view: a pop-up menu will appear with different settings.




### 12.4.1.1 *Cursor controls*

**Add cursor:** every pose in the 3D-view corresponds to 1 cursor. You can add this cursor – it will appear in the cursors-tab of the output window. **Do not forget to synchronize your cursor!** . Navigating the cursor in the cursors tab will show the 3D-perspective corresponding with that sample in the data file.

### 12.4.1.2 *Animation controls*

The animation controls configure the moving properties of the 3D-view.

**Amplified motion:** This option allows you to amplify small movements or rotations with a certain amplification factor (see "View controls" on page 57). If you open this menu-item, you'll see the sub-menu on the left:

Click on "Set moment" to take the current cursor position as reference point. **Every displacement and rotation with respect to the reference point will be amplified**, and not the position of the object with respect to the reference frame. Suppose my object frame is at 1000 mm from the origin of my reference frame: with an amplification factor of 20, my frame would be depicted at 20 meters from the reference frame! Let's say I take the previous sample as reference point, and let's assume the object moved over 0,7 mm since that previous sample. Due to the amplification, the object will show a motion of 14 mm. **The reference acts as temporary reference frame**.

To activate the amplification, click "switch on". If you now right-click, and choose "Amplified motion" again, you'll see that this option changed to "Switch off". Choosing this option stops the amplification. The motion of the frame will be expressed again with respect to the reference frame.

**Animation:** this option replays a specified sequence of the motion. If you open this menu item, you'll get the following possibilities:

**Set start point:** to take the current timestamp of the cursor as start point.
**Set endpoint:** to take the current timestamp of the cursor as endpoint of the animation sequence.
**Start animation:** to show the specified sequence at a very slow pace, allowing you to carefully study every detail of the motion. The motion will be repeated indefinitely. Should you right-click and choose "Animation" again, you'll see that this menu-item changed to
**Stop animation**: choosing this option will stop the animation.
**Save as AVI:** to export the animation as an *. AVI-file that you can view with any multimedia player.

**View:** to set the observer's viewpoint at 1 of 7 predefined viewpoints (top, bottom, left, right, front, back or isometric)

**Look from:** to set the observer's viewpoint equal to the **current** position of one of the objects in the current view (these objects are listed when opening the menu item). The current position is indicated by the cursor.

**Look at:** to set the observer's point of focus equal to the **current** position of one of the objects in the current view (these objects are listed when opening the menu item). The current position is indicated by the cursor.

#### *12.4.1.3 View controls*

You can pan the view by moving the mouse over the graph, holding down the Shift-key and the left mouse key: as long as the left button is down, the graph will follow the movements of the mouse.

You can rotate the view by moving the mouse over the graph, holding down the Ctrl-key and the left mouse key: as long as the left button is down, the graph will follow the movements of the mouse.

Rotating the mouse wheel will zoom in and out on a view.

If you hold down the keys Ctrl + Shift + left mouse button, and you move the mouse up and down, you can zoom in on the entire graph.

Some options for the visual representation are set with the "**Settings**" option after right-clicking in the 3D-view:

**General:**

- **Display axes**: if unchecked, the axes of the reference frame will disappear
- **Draw floor**: if checked, the checkered floor is drawn; if not, the floor is omitted
- **Perspective**: if unchecked, all drawings are drawn isometrically
- **Draw mode**: "Wire frame" just draws the outer edges so that objects appear transparent. "Filled" renders the objects, but no light effects. "Smooth" applies light effects
- **Up equals**: allows the user to select which axis of the reference frame should be drawn bottom-up (older versions always held Z+ as "up").

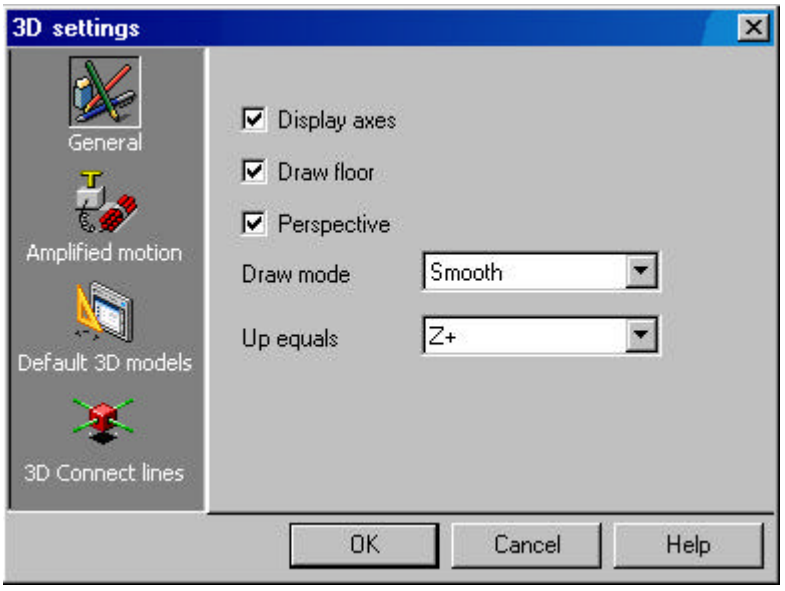



*Figure 38: 3D view settings window – general settings*

**Amplified motion:** entering values will amplify the translation or rotationalong the specified axis by the specified amplification factor. Amplification will not take place unless activated by the Animation controls on page 12.4.1.2.

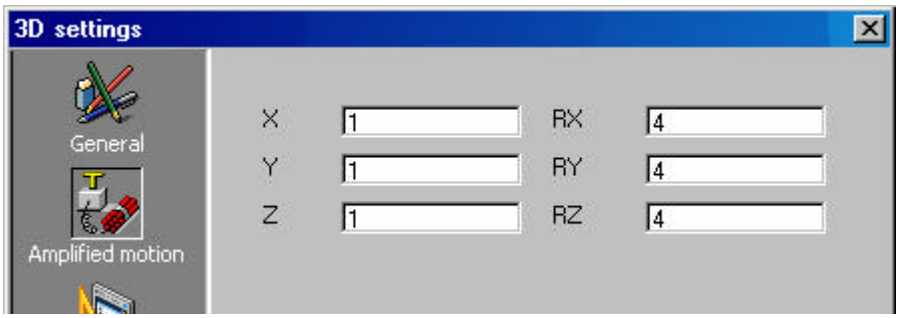



*Figure 39: 3D view settings window – general settings*

**Default 3D models:** this option allows the user to specify the default *.OBJ or *.DXF-file to be used for camera, 3D-object or 6D-object. *.DXF files will be shown in white.

**3D connect lines:** this menu controls wether or not lines are drawn between objects .

- **connect lines sequentially** : this option will draw a line between each object in the view, disregarding leds or 6D frames. The line starts at the first object, and ends at the last. E.g. if 3 leds and a 6D-frame are in view, the line will start at led 1, run through leds 2 and 3, and end in the 6D-frame.
- **use the following connections**: this allows the user to specify manually the lines to draw. Series are separated by commas; a discrete line between 2 points is indicated by a dash, a discrete line through consecutive points is indicated by a semicolon. E.g. “1:4 , 5-7” draws 2 lines: 1 line through led 1-2-3-4, 1 through led 5 and led 7.

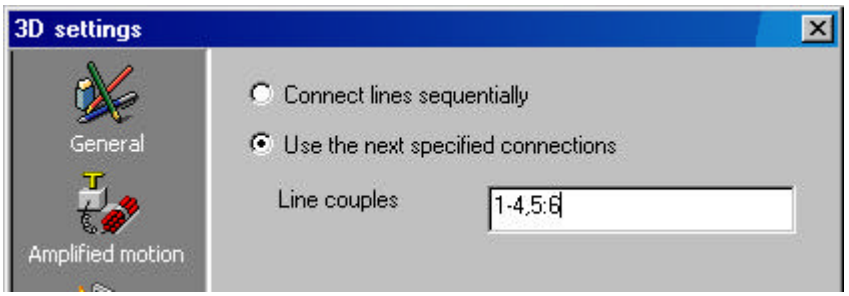



*Figure 40: 3D view settings window – 3D connect lines settings*

# 13. Exporting to Matlab

This chapter describes the interaction between DMM Modular and Matlab.
Should you encounter a problem, please refer to the troubleshooting chapter "Exporting to MATLAB" on page 59.

DMM Modular has the option to export data to Matlab through an internal, direct link.
This allows you to export measured data immediately and manipulate them interactively.

This option is only available in the DMM Modular 'Advanced' version.

The advantage over using modules here is the increased flexibility: the user can improvise, and execute calculations that were not foreseen. It is more intended for the "laboratory" stadium of problem solving.

Make sure Matlab is NOT started.
Choose "Project" – "copy to Matlab" from the main menu, or click on the "Matlab" icon on the icon bar:

Matlab will be started in the background.
Every data file, identification, … will be teleported to Matlab. In Matlab, each file is converted into a matrix. The order of columns is equal to the layout as seen in a spreadsheet view.

Use the Window task bar to bring Matlab to the foreground and type "Whos" you'll get a list of all the matrices.
**Warning !** if you are using long filenames, Matlab might shorten these names, so it might be difficult to find what matrix corresponds to what file. A quick comparison between a matrix plot and a 2D-graph in DMM Modular mostly solve the ambiguities.

If you are creating files AFTER Matlab was started, choose again "Project" – "Copy to Matlab", or click the Matlab icon again: this will copy all data files once again to Matlab, including the latest created files.

At the time of writing, the option to send modified matrices back to DMM Modular is not yet possible.

Do NOT close Matlab yourself: if you do so, DMM will not know that Matlab is no longer available. If you start Matlab manually, the communication link between those two no longer exists, and no transmitted data will arrive in Matlab. The only solution is to close DMM Modular and Matlab, start DMM Modular and repeat the procedure above.

# 14. Options in the main menu

Untill now, we've seen all possibilities to measure elements interactively. However, all these possibilities are explained on a 'only once' basis.
The main menu holds a number of features that allow us to organize and configure settings that are not likely to change often.

DMM Modular has the ability to detect new calculation modules. Every time such a module is found, it will be added to the menu. This manual will only describe the standard menu options.

The standard menu options are :

'**Project**' : this menu allows to organize measurement sessions into projects. It also gives the user acces to the deleted items bin, and the ability to return geometric elements to another Krypton software product.

'**View**' : this menu allows the user to customize the user interface: hide/show toolbars, adding/removing icons, status bars, …

'**Tools**': this menu allows the user to configure acquisition, file exports, …

'**Identify**': this menu starts GeoLoc for object identifications, and to measure frames that can be used as reference frames.

'**Script**': this menu holds all tools to record, play, or edit "scripts", macro's that can automate repetitive tasks.

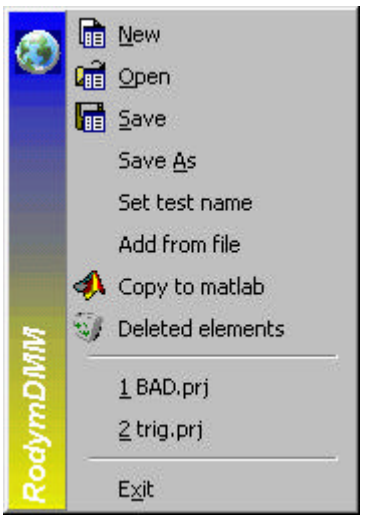
'**Window**': this menu allows the user to create new views, and to organize the different graph's ( tiling, cascading, … ).

'**Acquisition**': this menu holds all the tools to perform an acquisition, create frame data, and express this data in any other defined reference frame.

'**Help**': holds the "About"-box, and the on-line help function.

## 14.1. 'Project' menu

The project menu deals with three features: organizing projects, managing deleted elements and exporting elements to other Krypton products.




In general, DMM Modular sorts all measurements in projects. These are separate directories on the hard-disk. By storing geometric elements, the element window will not be flooded with elements that are outlived, or have got nothing to do with the current measurement context.

The user must not be aware where the project files are stored. DMM Modular stores this information for him. All the user has to do is open the appropriate project file. DMM Modular will fill the workspace window with the corresponding elements. From then on, it's just as if the user never stopped measuring.

When identifying objects, DMM Modular will set the appropriate project for GeoLoc.

The current project is denoted in DMM Modular's status bar : e.g. the project 'myFirst'

Should it be necessary to exchange elements between projects, DMM Modular can import element files from other projects.

Whenever an element gets deleted, its file is not yet physically removed, but the element's icon disappears from the workspace window. The project menu allows to restore elements, or to purge the 'Deleted items' bin.

Finally, The menu allows to export data files to Matlab as matrices.

We'll now discuss each option in detail :

**New :** creates a new project. When a new project is created, a new data direc-tory is created for storing the data files, and a project file ( extension ***.PRJ** ).
The user is prompted to enter a project name. While the user types the project name, the data directory will automatically grow. It is possible to change the data directory *after* the project name was entered.

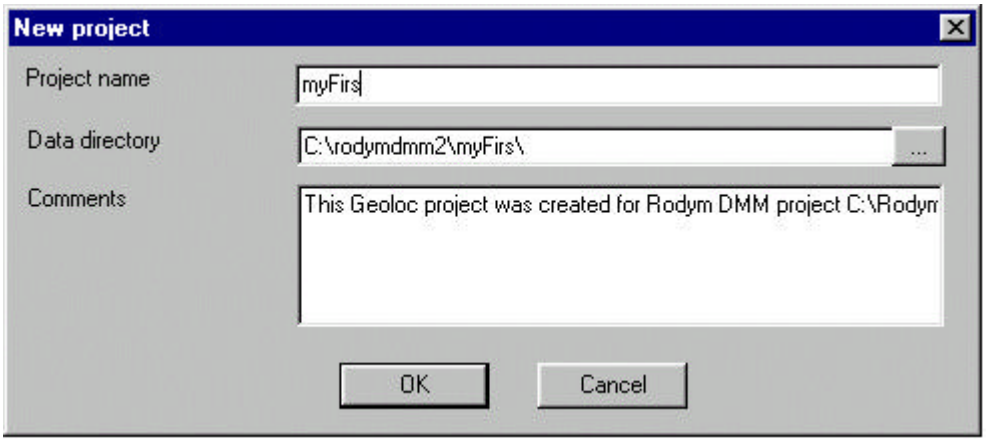



*Figure 41: 'New project' window*

Should the user change the project name after the data directory was set, the data directory will be reset to the default directory, extended with the project name.

Within this project directory, a separate GeoLoc directory will be created.

**Open :** select the project file ( project name with extension *.PRJ ). DMM Modular will fill the workspace window with all data files described in this project.

**Save** : stores the current project file.

**Save As** : will store the current elements in a new project. The pop-up window is the same as for 'New', but it will not be an empty project.

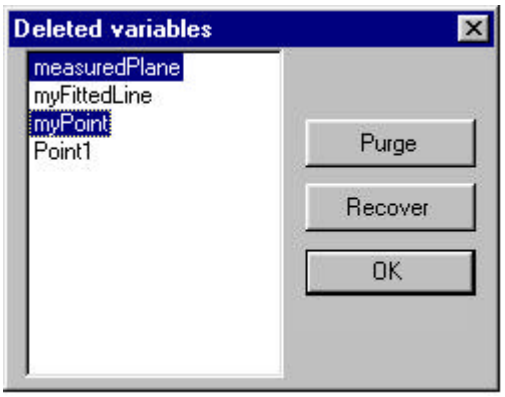
**Set test name** : creates or opens a test folder in the current project. See chapter "The 'test' level" on page 90.

**Add from file** : allows to add manually data files from other projects. All elements are stored on disk under Matlab format. Such file is of form 'elementName.MAT'. Select the Matlab file that corresponds to your element with the pop-up form.

**Copy to Matlab** : if clicked, this will open a Matlab session and copy all data files in the current project to Matlab matrices. When clicking this icon after Matlab was already started, it copies all the data files to Matlab again.
This function is only available in the DMM Modular 'Advanced' version.

**Deleted items** : choosing 'Deleted items' lists the elements that were deleted but not physically removed from the hard disk.
If you select a number of elements and click on 'recover', the elements are restored in the element window.
Clicking 'Purge' will physically remove the selected elements from the hard-disk.

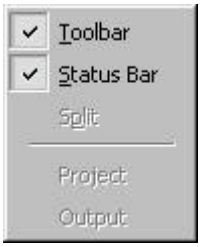
Clicking 'OK' returns you to the main menu.




**Exit** : closes DMM Modular

## 14.2. 'View' menu

The view menu is a small menu that allows the user to modify the user interface to his/her personal comfort.




Unchecking the 'Toolbar' will hide the icon bar below the menu
Unchecking the 'Status bar' will hide the status bar at the bottom of DMM Modular.
Unchecking the 'Output' will hide the output window
Unchecking the 'Project' will hide the workspace window

## 14.3. 'Tools' menu

The tools menu holds three configuration functions:

- options : sets configurations on measurements and calculations
- Customize : brings up a form that allows you to customize every item on the menu and icon bar
- Users : allows you to define users, and assign rights for operations.

### 14.3.1. 'Options'

**Options** : this function brings up a pop-up with settings sheets : "General", "CSV export" and "Acquisition". "General" configures for settings that apply to DMM Modular in total ; "CSV export" allows to configure the data export to ASCII files, depending on regional settings. "Acquisition" configures for acquisition-related settings.
**Every time a module is added, a new tab sheet will appear where the user can enter the settings for this new module**

#### 14.3.1.1 *General options*

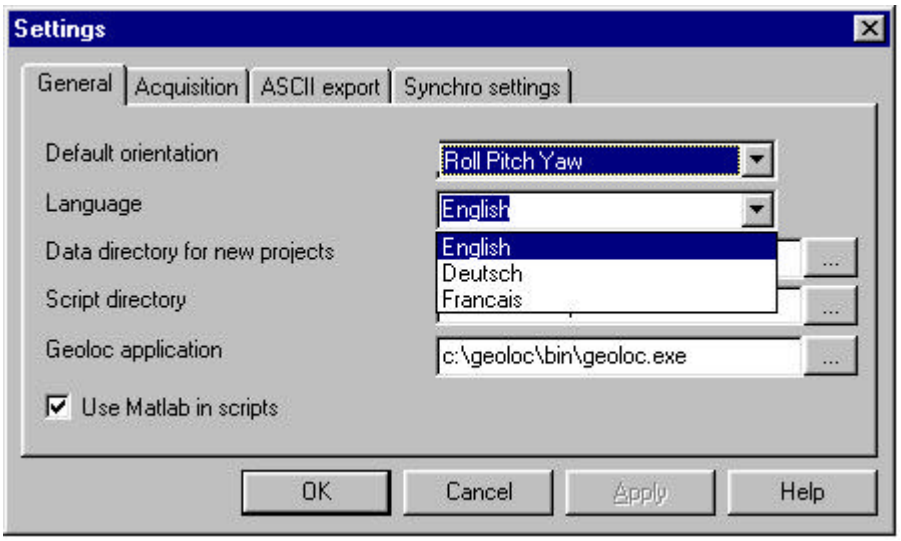



*Figure 42: General settings tab sheet*

**Default orientation** : this is the default orientation convention ( see chapter "Orientation conventions" on page 19.

**Language** : language of the user interface. The changes will be applied the next time the application is started.

**Data directory for new projects** : this is the default directory DMM Modular will create subdirectories in, every time a new project is created.

**Script directory** : default directory where DMM Modular will look for scripts.

**Geoloc application** : full path and name of the application that will be called when an object identification must be performed.

**Use Matlab in scripts** : if checked, DMM Modular scripts will be able to execute Matlab scripts.

#### 14.3.1.2 *ASCII Export options*

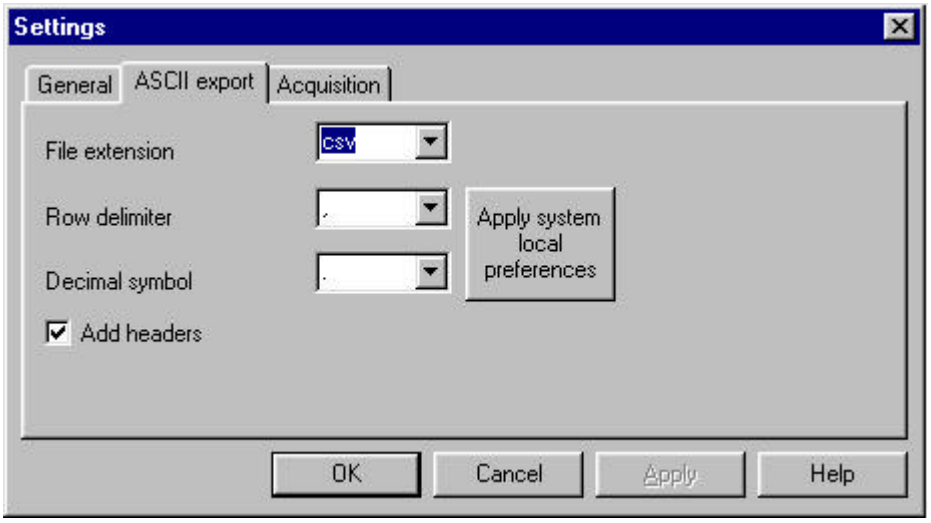



*Figure 43: 'ASCII export' settings tab sheet*

**File extension** : extension of exported data ASCII files. The extension can be chosen from predefined extensions (*.CSV or *.TXT ), or can be entered manually.

**Row delimiter** : this character will separate two data cells from each other. The delimiter can be chosen from the combo box, or can be entered manually

**Decimal symbol** : this character will separate the decimal fraction of a number from the rest of the value. The decimal symbol can be chosen from the combo box, or can be entered manually

**Apply system local preferences** : if you click this button, the software will search the operating system for the row delimiter and decimal symbol as set forth in the regional settings. Clicking this button is the best guarantee that applications such as Excel can read the exported files. However, if you wish to send the files to a colleague who uses different regional settings, you might have to specify row delimiter and decimal symbol manually.

**Add headers** : if you select this option, two lines with textual information will be inserted at the beginning of the file. The first line holds the name of each column; the second line holds the unit in which each column is expressed.

#### *14.3.1.3 Acquisition options*

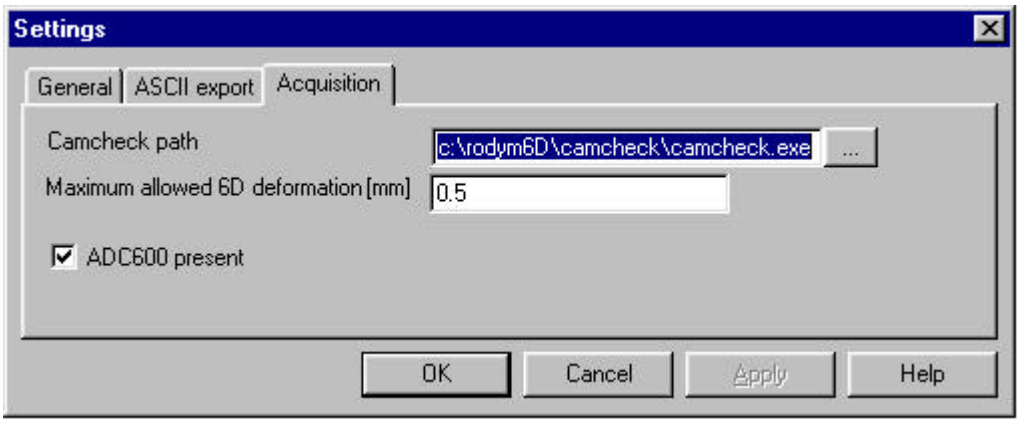



*Figure 44: 'Acquisition' settings tab sheet*

**CamCheck path** : full path and name of the application that can act as configuration program, and can check the environment (refer to the manual “Rodym® 6D system guide”) .

**Maximum allowed 6D deformation** : this is a quality factor for deriving frame motions from individual LED motions. If the calculation error exceeds this value, the frame will be considered invisible. ( invisible LED’s are projected at a very large distance, making it look as if the frame deformed enormously)

**Warm-up time** : the time (in seconds) in which the LED’s will warm up before the actual measurement starts.

**ADC600 present** : check this box if :
?? you have an ADC600 acquisition box connected
?? ÁND you’re using it ( measure at least 1 analog, digital or counter channel).
If not, the analog, digital and counter acquisitions could be erroneous, and your camera external synchronisation will not work.

### 14.3.2. 'Customize'

**Customize :** this menu brings up a form that allows you to customize every item on the menu and icon bar. The form has a number of tab sheets :

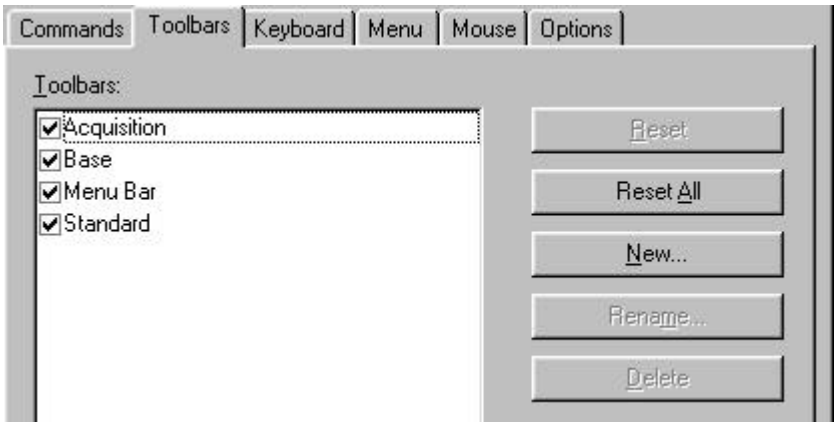



*Figure 45: 'Customize' tab sheet - toolbars*

**Toolbars** : this tab allows you to create new toolbars, hide existing toolbars, or deleting toolbars. It is suggested to create a personal toolbar to place all user-chosen icons on. When a new toolbar is created, this starts as a very small grey block. Drag it besides the icon bar.
Checking or unchecking a toolbar shows or hides the toolbar.

**Warning !** If you click 'Reset all', all menu entries of additional modules will be deleted! If you want to reset the toolbars, while retaining the module toolbars, go to the tab 'Menu' and click on 'Reset' for 'Default menu'.

**Commands:** this menu shows, per category, every possible menu item and its corresponding icon. Just select the icon of a function and drag it onto the icon bar. If you want to remove an icon, just click on it on the menu bar, drag it into the form while holding your left mouse-button down.

**Keyboard:** this menu allows to configure keystrokes as shortcuts for certain functions: Select a category; in the listbox 'Commands', a list of all functions for that category appear. Select the command you want to create a shortcut for. In the 'Description'-window, you'll get a description for the chosen command. Should a shortcut already be defined for this command, it will appear in the list-box 'Current keys'.
Click with the mouse in the text box labeled 'Press new shortcut key:'. Just press the key combination you'd like to use to activate the command : this text box will show the key combination you just pressed.
If this key combination was not yet assigned, a message will appear below the text box. Click on the 'assign' – button to confirm the assignment. The shortcut will be added to the 'Current keys' listbox.
If a key combination was already assigned, a message will appear indicating which command is already linked to that shortcut. It will be impossible to re-assign this shortcut. If you wish to remove a shortcut, select it in the 'Current keys' listbox and click on 'Remove'.

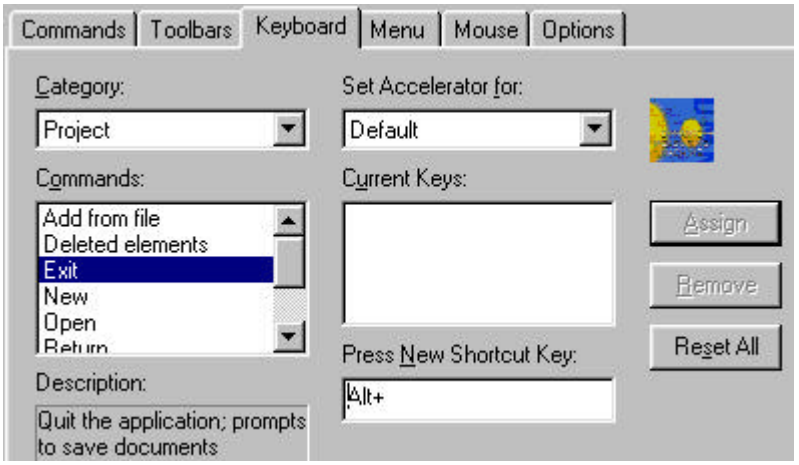



*Figure 46: 'Customize' tab sheet - keyboard*

It is suggested not to change the default settings for the following 3 menu's:

**Menu** : this menu allows to configure some menu animations or context menu's.

**Mouse**: allows to configure the double-click command of the mouse.

**Options**: allows to configure the Windows Screen Tips and the Windows 2000 look.

### 14.3.3. 'Users'

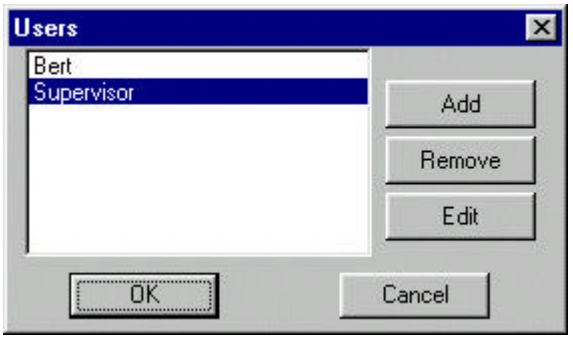
**Users :** this menu brings up a form that allows you to limit the number of manipulations a user can execute.

The main idea behind DMM Modular's user management, is NOT to prevent a user from abusing the system, but to limit the overhead of buttons and features : a mean and lean tool for measuring.

The default user is "Supervisor" : this user has all rights.

How to create a new user and assign his rights ?
Add a new user (see below), and copy the settings from the supervisor, e.g. *newUser*.
Next, log out and log in as *newUser* .
Modify the user interface (add/remove icons, remove items from the main menu), and only leave those items that *newUser* should have acces to ( see chapter " 'Customize' " on page 65). **Don't forget to remove the "Tools" –"Customize" and "Tools" –"Users" options !**
If you log out again, *newUser* will only have acces to the last menu items and icons that were present at log-out time.

When clicking "Users", the following form appears :



**Add:** adds a new user. If you click this, a small pop-up appears, asking you to enter a user login and password for the new user.
If you like, you can also copy the settings from a previously defined user.

**Remove:** erases the definition of a user.

**Edit:** allows you to change a user name and/or password. You can't copy the settings from another user any more.

## 14.4. 'Identify' menu

starts GeoLoc. Please refer to the Geoloc user manual.

**Note:** if DMM starts to measure, and Geoloc is still open, Geoloc will be closed, to prevent two applications from attempting to acces the camera simultaneously.

## 14.5. 'Script' menu

Although the macro's themselves are fully described in chapter "Scripting – writing and using macro's" on page 71, we'll list the menu possibilities here already.
**New :** new script

**Open :** open an existing script

**Record:** when this option is chosen, the macro 'teach' mode is toggled. If the 'teach'-mode is active, the icon will look as if it is pushed in. From this moment on, every action performed is recorded into a macro and can be replayed later.
When activating the 'teach'-mode, DMM Modular asks in which script to dump the actions. This file is an ASCII-file, and can be edited with any editor. If the file already exists, DMM Modular will ask if you want to overwrite the file, or if you want to append to it.

**Save :** save a script under its original name

**Save as :** save script under a different name

**Print :** sends the current script to the default printer.

**Close :** closes the current script

**Run** : run a previously recorded macro. When clicking the function, it will ask for a macro to run. Macro's have the file extension "*.SCRIPT".

Clicking "Run" will play a script from line 1; if you want to run a script from a specific line, open the script, position the text cursor on the first line to execute,and right-click with the mouse ; from the context menu, select "Run from current line".

**Edit** : this script opens the macro editor ( see chapter "Scripting – writing and using macro's" on page 71)

**Interrupt** : when a script is running, this will interrupt the active script and return program control to the DMM Modular user interface.

**Clear output** : clears the output window.

**Cut :** move a portion of text to Windows' clipboard

**Copy :** copy a portion of text to Windows' clipboard

**Paste :** insert the clipboard contents

**Find :** find the occurrence of a string

**Replace :** replace the occurrence of a string with another string

**Show functions :** this shows a structured overview of all the available functions for scripts :

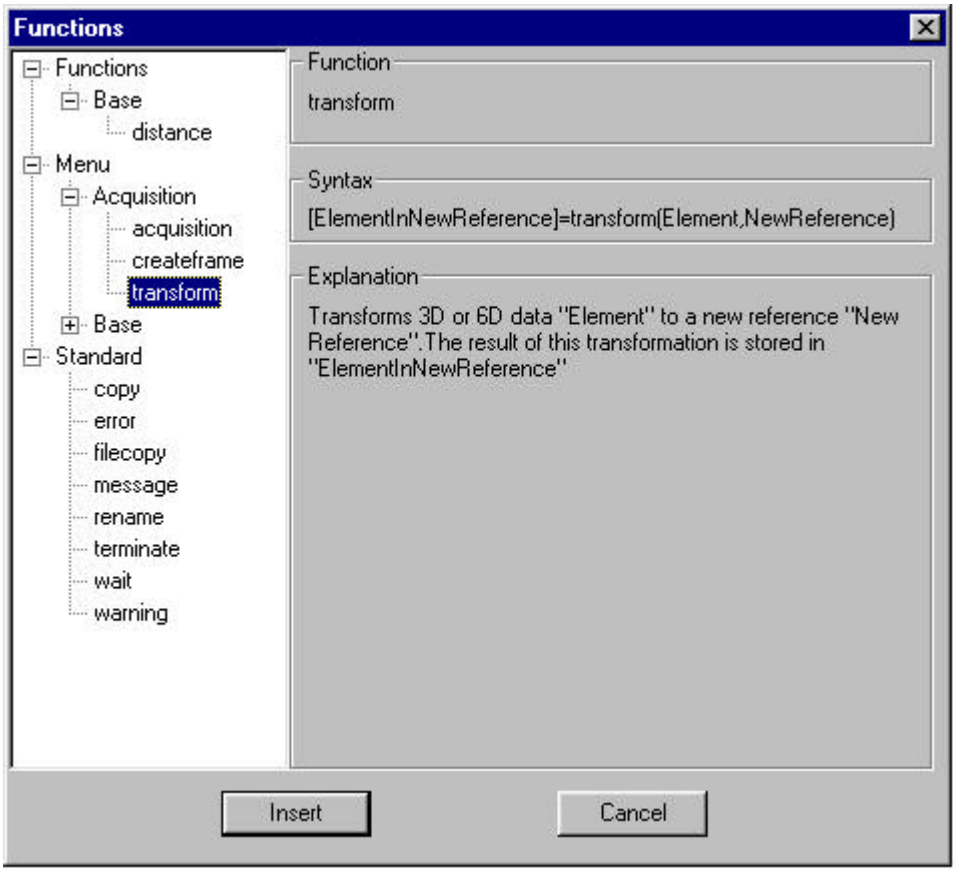



*Figure 47: script editor - the function browser*

If you select a function from the tree, on the right will appear the function name, syntax, and an explanation on the function.

By clicking on 'Insert', the function will be inserted in the current script wherever the text cursor was positioned at the time of insertion. The inserted line will contain dummy parameters, which the user can replace with actual parameters.

Note that it is also possible to include previously defined scripts.

When a function is inserted, it is inserted with dummy parameters. These parameters must be replaced by the user with their 'real-life' counterparts.

**Undo :** neutralize last edit action

## 14.6. 'Acquisition' menu

**Measure:** this function has been described in chapters "Configuring a measurement" on page 26 and chapter "Measuring single LED's or objects" on page 32.

**Create 6D data :** this function is described in chapter "The function 'Create 6D data'" on page 34.

**Transform:** this function is described in chapter "The function 'Transform'" on page 35.

## 14.7. ‘Base’ menu

**Calculate velocity:** this function has been described in chapter “The function ‘Calculate velocity’” on page 37.

**Change orientation convention :** this function has been described in chapter “The function ‘Change orientation convention’ on page 40.

**Calculate point in 6D frame :** this function has been described in chapter “The function ‘Calculate point in 6D frame’” on page 41.

**Offset :** this function has been described in chapter “The function ‘Offset’” on page 36.

**Fit polynome :** this function has been described in chapter “The function ‘fit polynome’” on page 42.

**Overshoot :** this function has been described in chapter “ The function ‘overshoot’” on page38.

**Find maximum :** this function has been described in chapter “The function ‘Find maximum’” on page39.

**Find minimum :** this function has been described in chapter “The function ‘Find minimum’” on page39.

**Calculate Mean :** this function has been described in chapter “The function ‘Calculate mean’” on page39.

**Standard deviation :** this function has been described in chapter “The function ‘Calculate std’” on page40.

**Filter :** this function has been described in chapter “The function ‘Filter’” on page 44.

**Frequency spectrum :** this function has been described in chapter “The function ‘Frequency’” on page 44.

**Join measurements :** this function has been described in chapter “The function ‘Join measurements’” on page 45.

**Distance :** this function has been described in chapter “The function ‘Distance’” on page 37.

## 14.8. ‘Help’ menu

This menu has two functions:

**Help** : activates the Windows on-line help function on DMM Modular

**About DMM Modular** : returns a splash screen with DMM Modular’s version number and build number. Always communicate these numbers when reporting problems to Krypton.

# 15. Scripting – writing and using macro's

Scripting is a very powerfull utility of DMM Modular.
Scripting allows you to record or program a set of measurements. This macro can be replayed an infinite number of times.
In a script, clarifying messages can be sent to the operator: this means that one person can write scripts, distribute them to multiple measurement stations, where an operator just follows screen instructions.

DMM Modular scripts can execute a variety of functions :

- they can execute menu actions
- they can execute other DMM Modular scripts
- they can execute Matlab scripts

DMM Modular scripts are ASCII-files: they can be read, modified or saved by any text editor.
DMM Modular's own script editor however has some utilities that make it a very handy tool when dealing with scripts.

The total length of a script file name (plus his path) can not exceed 128 characters ! If it does, the script will not be run.

We'll start with explaining how to write an easy, top-level script. Next, we'll go into how to use scripts within scripts

## 15.1. Writing a single-level script

DMM Modular scripts can be created in two ways :

- they can be written off-line using a dedicated DMM Modular editor
- they can be teached: DMM Modular records all the operations performed by the user

Both methods have their advantages and drawbacks:

- off-line writing has the greatest flexibility, but requires the user to be familiar with the DMM Modular script syntax. Unexperienced users can easily make mistakes this way
- teaching is the easiest: the user is not confronted with DMM Modular's syntax. But this means that a number of functions (such as the message( ) function ) are not accessible in teach-mode. Scripts are sometimes longer, due to rename-operations, …

The advised method is a mix of the two: record the first draft of the script using the 'teach'-mode ; then, open the script in the DMM Modular editor (or any other editor), filter out the overhead manually, and add message( )-functions at will.

When reading the following chapters, please bear in mind that choosing an option from the main menu can also be done by clicking the corresponding icon on the icon bar. See chapter "'Script' menu" on page 67 for these icons.

### 15.1.1. Writing a script

This chapter will clarify how we teach a script.
Go to the main menu and choose 'Script' – 'Record'
DMM Modular will ask for a script file name in the current project directory. Enter a suitable file name. Make sure the extension is 'script'.

Start measuring/calculating data files with the probe as if no script was recorded. In the background, every action is recorded. You'll see that the record-icon looks pushed into the user interface.

When you are done, go again to the main menu and choose ‘Script’ – ‘Record’. The icon will click out of the menu.

Now choose ‘Script’ – ‘Open’. Enter the filename under which you recorded your previous program. The DMM Modular editor will open the script :

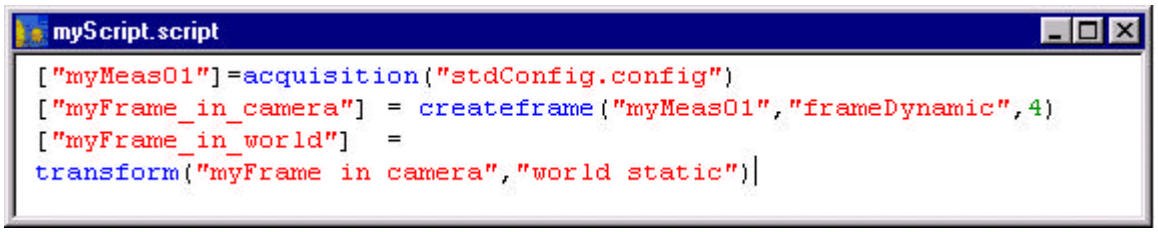



*Figure 48: Scripting - recording a script*

As you can see in the example above, no messages are programmed to be sent. Therefore we can manually edit the script, to add some messages and some more calculations. The result is the following :

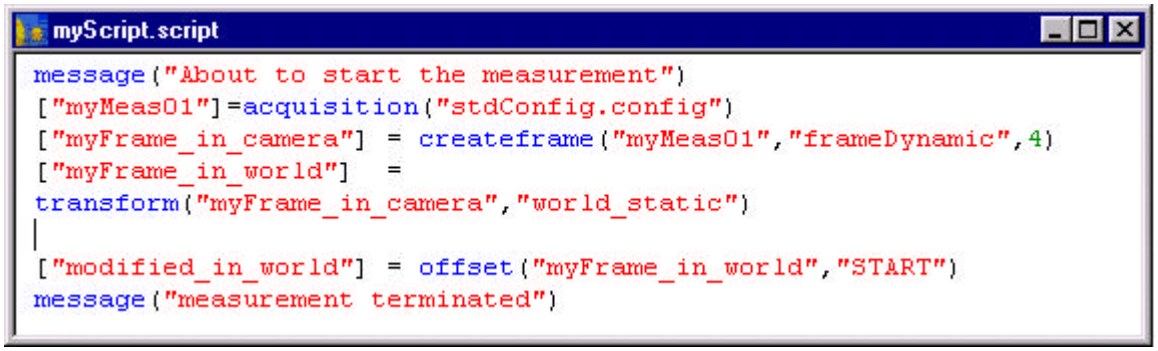



*Figure 49: Scripting - editing a script*

This script can be saved and can now be run.
The functionality the DMM Modular script editor is explained in chapter “‘Script’ menu” on page 67.

If the script encounters a function that it does not recognize, it will try to execute it in a Matlab environment: this means that you can program Matlab functions or Matlab *.m-file names in a script.

### 15.1.2. The script syntax

The DMM Modular scripting syntax is very similar to the Matlab syntax :

- Both variables and function calls are case sensitive
- Most functions are of the form [element] = function(parameter). Return parameters are put between square brackets ; parameters are put between normal brackets.
- It is possible to use multiple parameters: parameters are separated by commas
- It is possible to return multiple return values: return values are separated by commas
- Elements can be put between double-quotes ( “ “ ) or not. E.g. [myPoint] or [“myPoint”] – although it is suggested to place them always between double quotes.
- Variables **have** to be put between double-quotes if their name contains spaces, or if they are constants, e.g. [“my Point”] or [“0.5”]
- Instead of writing variables, variables can be dragged from the data window and dropped behind the opening brace of the parameter list

- If in the function selector a function syntax is depicted as {element1, element2}, than this means that one of the elements between braces should be picked.
- If in the function selector a function syntax is depicted as <element>, than this means it is optional. Optional elements should **NOT** be written in the script, and should not be put between the <>-signs. E.g.:

*Right :*
*[“output”]=velocity(input)*
*[“output”]=velocity(input,”scalar”)*
*["modified_in_world"] = offset("myFrame_in_world","START")*

*Wrong:*
*[“output”]=velocity(input,<”scalar”>)*
*["modified_in_world"] = offset("myFrame_in_world",{"START"} )*

## 15.2. Using scripts within scripts

A script can be regarded in two ways : as a fixed set of operations, or as a function that calculates one specific sort of data.

The single-level script (see chapter “Writing a single-level script” on page 71) usually is a fixed set of operations.
When using nested scripts, the top-level script also is just a fixed set, but the branches can be of two sorts.

We’ll clarify the difference with an example. Suppose I have a calculated a data file “measurement1” and “measurement2” . Suppose I have three scripts, TOPLEVEL.script, FOO.script and BAR.script :

```
=== top of file  ======================
% this is script TOPLEVEL.script
% this executes the script FOO.script
FOO
% this executes the script BAR.script
[“output3”]=BAR(“measurement1”)
=== Bottom of file ===================
```

The script FOO is a fixed set of operations. All the parameters it uses and generates are global, i.e. they are created and known within DMM Modular.

```
=== top of file  ======================
% this is script FOO.script
[“output1”]=velocity(“measurement1”)
[“output2”]=velocity(“measurement2”)
=== Bottom of file ===================
```

The script BAR is a function: The first line states the syntax and return parameters to call this function. The first comment block (starting with percentage signs) will return information in the user interface, should you call this function without specifying parameters. All data files created within a script have a only a local scope, i.e. they are only known to the script, and only live as long as the script is executed. The variable *Velocity* will not appear in DMM Modular, since it is created locally ;

```
=== top of file  ======================
function [result] = BAR (“local Parameter”)
% BAR calculates the acceleration of your input parameter

[“velocity”]=velocity(“local Parameter”)
```

["result"]=velocity("velocity")
=== Bottom of file ==========

**Important notes :**

- Do **NOT** leave blanks between the function name and the first bracket: this will cause an error
- Make sure the script file name and the function name differ somewhere, otherwise the script parser will detect recursive functions

*Right :*
*=== top of file BAR.SCRIPT ===========*
*function [result] = BAR_("local Parameter")*
*["output"]=velocity(input)*
*==========*

*Wrong:*
*=== top of file BAR.SCRIPT ===========*
*function [result] = BAR_ ("local Parameter")*
*["output"]=velocity(input)*
*==========*

*=== top of file BAR.SCRIPT ===========*
*function [result] = BAR("local Parameter")*
*["output"]=velocity(input)*
*==========*

In case you would run a function script as top-level script, or running it without parameters, DMM Modular will show you a pop-up box, asking for the required in- and output parameters. The function BAR( ) would evoke this pop-up box :

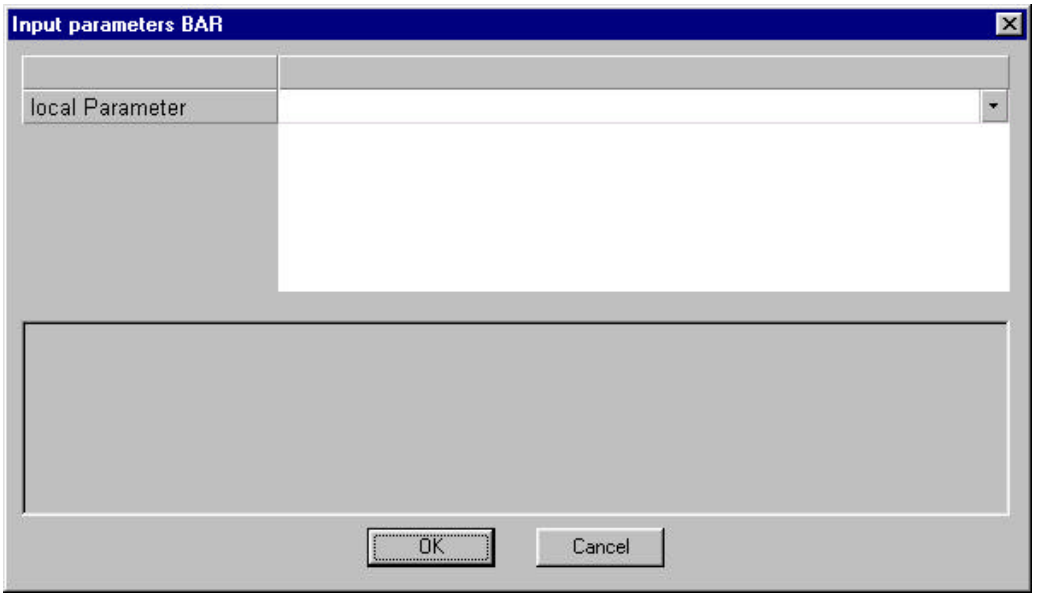



*Figure 50: script parameter pop-up form*

These parameters can be selected from combo boxes in the pop-up, or can be drag-and-dropped from the data window.

# 16. Script function reference

The script syntax for each function is explained in this chapter.

All the functions are listed alphabetically – but the following summary groups them by functionality. There is also a reference to the chapter of their interactive counterpart (if applicable) which were described in the previous chapters.

Acquisition & basic calculation

| Function | description | reference | Page |
|---|---|---|---|
| Acquisition( ) | Measures using settings from a file | 9 & 10 | 77 |
| createframe( ) | Calculates 6D-frames from led data | 11.1.2 | 79 |
| transform( ) | Transforms an element into another reference | 11.1.3 | 88 |

Advanced calculation

| Function | description | reference | Page |
|---|---|---|---|
| Compensatelinear phasefilter() | Compensates measurements for time lags due to an external analog filter (e.g. anti-alias filter) | - | 79 |
| calculatemean( ) | Calculates mean value for each column in a file | 11.2.5.3 | 78 |
| calculatestd( ) | Calculates standard deviation for each column in a file | 11.2.5.4 | 78 |
| distance( ) | Calculates 3D-distance between 2 elements | 11.2.2 | 80 |
| FFT( ) | Performs a Fourier analysis on a data element | 11.2.11 | 81 |
| filter( ) | Applies a low- or high-pass digital filter | 11.2.10 | 82 |
| findmaximum( ) | Returns the maxima's of a data element | 11.2.5.1 | 82 |
| findminimum( ) | Returns the minima's of a data element | 11.2.5.2 | 83 |
| offset( ) | Subtracts an offset for columns of an element | 11.2.1 | 85 |
| Orientationchange( ) | Changes angular conventions of 6D-frames | 11.2.7 | 85 |
| overshoot( ) | Calculates the overshoot of an element | 11.2.4 | 86 |
| pointinframe( ) | Calculates 3D-motions of other points of a dynamic frame | 11.2.8 | 86 |
| polyfit( ) | Fits a polynome on a correlation of elements | 11.2.9 | 87 |
| joinmeasurement( ) | Joins 2 measurement files into one | 11.2.12 | 84 |
| velocity( ) | Derives an element once versus time | 11.2.3 | 89 |

File manipulations

| Function | description | reference | Page |
|---|---|---|---|
| addfromfile( ) | Creates an element based on an imported file | 14.1 | 77 |
| copy( ) | Copies an element to another element | - | 79 |
| delete( ) | Deletes an element from the workspace | - | 79 |
| export( ) | Exports an element into a text or binary file | 11.2.6 | 81 |
| filecopy( ) | Copies a file from one location to another | - | 82 |
| globaladd( ) | Adds one or more elements to globals list | 17.2 | 83 |
| globalemptylist( ) | Clears the globals list | 17.2 | 83 |
| rename( ) | Renames an element | - | 87 |
| saveall( ) | Forces all elements to be saved on disk | 14.1 | 87 |
| increasetestname( ) | Creates a new test with incremented index | 17.1 | 84 |
| settestname( ) | Sets the testname to be used from now on | 17.1 | 88 |

Script control functions

| Function | description | reference | Page |
|---|---|---|---|
| error( ) | Yields an err message in the message window | - | 80 |
| message( ) | Yields an info message in the message window | - | 84 |
| clearmemory( ) | Clears the memory from unused files | - | 78 |
| terminate( ) | closes DMM | - | 88 |
| wait( ) | Stalls execution untill time expiry or key-press | - | 89 |
| waitmessage( ) | | - | 89 |
| warning( ) | | - | 89 |

Miscellaneous

| Function | Description | reference | page |
|---|---|---|---|
| identify( ) | Starts the GeoLoc application | 8 | 83 |

## 16.1. acquisition( )

| | | |
|---|---|---|
| **Syntax** | [Result] = acquisition(ConfigFile,Freq) | |
| **Input** | ConfigFile STRING<br>Freq FLOAT | Optional<br>Optional |
| **Output** | Result STRING | Optional |
| **Description** | This function executes a measurement of led's and/or analog channels, digital inputs and counter/encoders. *ConfigFile* is the full path and file name of the acquisition settings file: all acquisition settings (timing, frames or leds, strober ports, channel labels, …) are read from *ConfigFile* . If a measurement frequency *Freq* is specified, this will override the frequency specified in *ConfigFile*. If the output file name *Result* is specified, this will override the file name specified in *ConfigFile.* If *ConfigFile* is not specified, the parameters from the previous measurement are used. | |
| **Example(s)** | acquisition("C:\settings\acq01.settings")<br>acquisition("C:\settings\acq02.settings",100.00000)<br>["data"]=acquisition("C:\settings\acq03.settings")<br>["data"]=acquisition("C:\settings\acq04.settings",150.00000) | |
| **See also** | | |

## 16.2. addfromfile( )

| | | |
|---|---|---|
| **Syntax** | [Element1, …] = addfromfile(File1, …) | |
| **Input** | File1 STRING | Required |
| **Output** | Element1 STRING | Optional |
| **Description** | This function adds an element *Element1* (or list of elements) into DMM based on the full path and file name of the binary file *File1* (or list of binary files). If *Element1* is not specified, an element name will be generated based on the name of the file. If multiple files or elements are specified, the number of files and elements must correspond.<br>If the binary file has his *.INFO-file in the same directory, DMM will import this file as well; if not, all channels in the new imported element will get default channel names. | |
| **Example(s)** | addfromfile("Z:\transfer\meas01.mat")<br>addfromfile("Z:\trf\m01.mat","Z:\trf\m02.mat")<br>["element01"]=addfromfile("Z:\transfer\meas01.mat")<br>["el01","el02"]=addfromfile("Z:\trf\m01.mat","Z:\trf\m02.mat") | |
| **See also** | Export( ) | |

## 16.3. calculatemean( )

| | | |
|---|---|---|
| **Syntax** | [Result] = calculatemean(Data, Commentline) | |
| **Input** | Data STRING<br>Commentline STRING | Required<br>Optional |
| **Output** | Result STRING | Required |
| **Description** | This function calculates the mean value of each column of element *Data* (except for the first column, since this is a time or index column). This row with mean column values is appended as the new last row of the element *Result*. If a comment string *CommentLine* is specified, this will be inserted in the last column of the appended row (last column means: the column behind the last numerical value). | |
| **Example(s)** | ["mean\test01"]=calculatemean("Position6D\staticTest")<br>["mean\test01"]=calculatemean("Position6D\staticTest", "this is the second row with mean values") | |
| **See also** | Calculatestd( ) | |

## 16.4. calculatestd( )

| | | |
|---|---|---|
| **Syntax** | [Result] = calculatestd(Data, Std_scaling, Commentline) | |
| **Input** | Data STRING<br>Std_scaling INTEGER<br>Commentline STRING | Required<br>Required<br>Optional |
| **Output** | Result STRING | Required |
| **Description** | This function calculates the standard deviation (?) of each column of element *Data* (except for the first column, since this is a time or index column). This row with mean column values is appended as the new last row of the element *Result*. If the parameter *Std_scaling* is 0, ? is calculated with (n-1) ; if *Std_scaling* is non-zero, ? is calculated with (n).<br>If a comment string *CommentLine* is specified, this will be inserted in the last column of the appended row (last column means: the column behind the last numerical value). | |
| **Example(s)** | ["std\test01"]=calculatemean("Position6D\staticTest")<br>["std\test01"]=calculatemean("Position6D\staticTest", "this is the second row with mean values") | |
| **See also** | Calculatemean( ) | |

## 16.5. clearmemory( )

| | | |
|---|---|---|
| **Syntax** | clearmemory(Time) | |
| **Input** | Time INTEGER | Optional |
| **Output** | | |
| **Description** | All measured and calculated files are kept in memory – but a full memory degrades performance quite strongly. This function will clear the memory from all elements that are not currently locked. If *Time* is specified, only elements that remained unused for *Time* seconds will be cleared from memory. | |
| **Example(s)** | clearmemory( )<br>clearmemory(120) | |
| **See also** | | |

## 16.6. compensatelinearphasefilter( )

| | | | |
|---|---|---|---|
| **Syntax** | [Result]=compensatelinearphasefilter(Data, nofSamples) | | |
| **Input** | Data<br>nofSamples | STRING<br>INTEGER | Required<br>Required |
| **Output** | Result | STRING | Optional |
| **Description** | Analog filters (such as anti-alias filters) delay the measured analog signals with respect to the physical analog signals. In reality, this time lag is expressed in a rounded number of samples (nofSamples = time lag x frequency). This function will shift the analog measurements *Data* forward in time over *nofSamples* samples, so it is in phase again with the physical analog values. *Result* is the result file in which the analog data will be saved. If *Result* is not specified, *Data* will be replaced by the shifted data. | | |
| **Example(s)** | compensatelinearphasefilter(“Acquisition\analog\Test01”, 7)<br>[“Acquisition\analog_sync\Test01”] =<br>compensatelinearphasefilter( “Acquisition\analog\Test01”, 7) | | |
| **See also** | | | |

## 16.7. copy( )

| | | | |
|---|---|---|---|
| **Syntax** | [Result] = copy(Data) | | |
| **Input** | Data | STRING | Required |
| **Output** | Result | STRING | Required |
| **Description** | This function copies the element *Data* to a new element *Result*. | | |
| **Example(s)** | [“Position6D\File_B”] = copy(“Position6D\File_A”) | | |
| **See also** | | | |

## 16.8. createframe( )

| | | | |
|---|---|---|---|
| **Syntax** | [Result, Residu] = createframe(Data, Identification, ledNr) | | |
| **Input** | Data<br>Identification<br>ledNr | STRING<br>STRING<br>INTEGER | Required<br>Required<br>Required |
| **Output** | Result<br>Residu | STRING<br>STRING | Required<br>Optional |
| **Description** | This function creates a 6D element *Result* based on the measured led data *Data* and the dynamic frame *Identification* . *ledNr* is the sequence number of the first led of the dynamic frame in the measured led data.<br>The 6D element *Result* will be expressed in the same reference frame as *Data* .<br>If *Residu* is specified, a new element will be created that holds for each led the residual error ( = error between measured led position and nominal led position in the dynamic frame). | | |
| **Example(s)** | [“Pos6D\object”] =<br>createframe(“Acquisition\camera\ledsOnObject”, “DynObject”, 4)<br>[“Pos6D\object”, “ResidualErr\object”] =<br>createframe(“Acquisition\camera\ledsOnObject”, “DynObject”, 4) | | |
| **See also** | | | |

## 16.9. delete( )

| | | | |
|---|---|---|---|
| **Syntax** | delete(Element1, …) | | |
| **Input** | Element1 | STRING | Required |

| | | |
|---|---|---|
| **Output** | | |
| **Description** | This function deletes the element *Element1* from the workspace.<br>At least 1 element name must be specified.<br>Multiple element names are accepted; wildcards are not . | |
| **Example(s)** | Delete("Pos6D\object")<br>Delete("Pos6D\object", "ResidualErr\object") | |
| **See also** | | |

## 16.10. distance( )

| | | |
|---|---|---|
| **Syntax** | [Result] = distance(Element1, Element2, LedIndex1, ledIndex2) | |
| **Input** | Element1 STRING<br>Element2 STRING<br>LedIndex1 INTEGER<br>LedIndex2 INTEGER | Required<br>Required<br>Required<br>Required |
| **Output** | Result STRING | Required |
| **Description** | This function calculates for every timestamp the 3D-distance ( D = ?(?x² + ?y² + ?z²) ) between *Element1* and *Element2* . If *Element1* is an element that holds point positions (leds), *LedIndex1* specifies which will be used; otherwise, *LedIndex1* should be 0. If *Element2* is an element that holds point positions (leds), *LedIndex2* specifies which will be used; otherwise, *LedIndex2* should be 0.<br>To get the distances in X, Y and Z of the reference frame, use the function offset( )<br>To get the distances in X, Y and Z of one of the two objects, use the function transform( ) | |
| **Example(s)** | ["Res\dist"]=distance("Pos6D\objectA", "Pos6D\objectB", 0, 0)<br>["Res\dist"]=distance("Acquisition\camera\test01", "Pos6D\objectB" ,2,0)<br>["Res\dist"]=distance("Pos6D\objectB", "Acquisition\camera\test01", 0, 2)<br>["Res\dist"]=distance("Acquisition\camera\test01", "Acquisition\camera\test02" ,2,4) | |
| **See also** | Offset( ) , Transform( ) | |

## 16.11. error( )

| | | |
|---|---|---|
| **Syntax** | error(ErrMessage) | |
| **Input** | ErrMessage STRING, FLOAT or INTEGER | Required |
| **Output** | | |
| **Description** | This function generates an error condition that will display the string *ErrMessage* in the DMM message window. **Be carefull !** Any action, associated with error conditions, will be executed as well ! (e.g. setting bits of the synchro module high, …) | |
| **Example(s)** | error("This is an error message")<br>error(5.023)<br>error(5)<br>error(myVariable) | |
| **See also** | message( ), warning( ) | |

## 16.12. export( )

| | | |
|---|---|---|
| **Syntax** | [FileName]=export (Element,Type) | |
| **Input** | Element STRING<br>Type STRING CONSTANT {“ASCII”,”MATLAB”} | Required<br>Required |
| **Output** | FileName STRING | |
| **Description** | This function exports the element *Element* from the workspace into the physical file *FileName* . Dependant on the value of *Type* , *FileName* is either an ASCII-file (*Type* = “ASCII”), or a binary, Matlab-compatible file ( *Type* = “MATLAB”). For an ASCII export, the user can specify the file extension, field separator and decimal symbol in “Tools” – “Options” – “ASCII Export”. For a Matlab-export, a *.info-file will be generated too for *Element* , describing the contents of each column. (this makes a later import with addfromfile( ) easier).<br>Special run-time variables allow to modify the result file name during script execution:<br>?? *project* : will be replaced by the current project name at run-time<br>?? *test* : will be replaced by the current test name at run-time – if the test name is not empty<br>?? **** : unique index number with the same number of digits as there are asterisks<br>?? *first* : extracts the first number in the variable name<br>?? *last* : extracts the last number in the variable name | |
| **Example(s)** | [“C:\Docs\test.csv”]=export(“Position6D\ObjectA”,”ASCII”)<br>[“C:\Docs\test.mat”]=export(“Position6D\ObjectA”,”MATLAB”)<br>Suppose the project name is “ABC” and the test name is “XYZ” :<br>[“C:*project**test*\test.csv”]=export(“ObjectA”,”ASCII”) will yield C:\ABC\XYZ\test.csv<br>Suppose this is the 105th execution of this script line :<br>[“C:\test****.csv”]=export(“ObjectA”,”ASCII”) will yield C:\test0105.csv<br>[“C:\test*first*.mat”]=export(“ObjA_mes2_run3”,”MATLAB”) will yield C:\test2.mat<br>[“C:\test*last*.mat”]=export(“ObjA_mes2_run3”,”MATLAB”) will yield C:\test3.mat | |
| **See also** | addfromfile( ) | |

## 16.13. FFT( )

| | | |
|---|---|---|
| **Syntax** | [Result] = FFT (Element) | |
| **Input** | Element STRING | Required |
| **Output** | Result STRING | Required |
| **Description** | This function performs a Fourier analysis on each data column of element *Element* . The function will yield the amplitude for each frequency in the range [0, F/2] ( F being the sample frequency) , scaled to 1. ( the sum of all amplitudes is 1 ). The amplitude for each frequency will be stored in *Result*. | |
| **Example(s)** | [“Frequency\Result”]=FFT(“Position6D\ObjectA”) | |
| **See also** | | |

## 16.14. filecopy( )

| | | |
|---|---|---|
| **Syntax** | filecopy (Source, Destination) | |
| **Input** | Source STRING<br>Destination STRING | Required<br>Required |
| **Output** | | |
| **Description** | This function copies the physical file *Source* to the physical file *Destination*. Source and destination should hold full path and file names. | |
| **Example(s)** | filecopy("C:\Results\report.xls", "D:\transfer\data.xls") | |
| **See also** | | |

## 16.15. filter( )

| | | |
|---|---|---|
| **Syntax** | [Result] = Filter (Element, Type, Poles, Frequency) | |
| **Input** | Element STRING<br>Type STRING CONSTANT {"lowpass", "highpass"}<br>Poles INTEGER<br>Frequency FLOAT | Required<br>Required<br>Required<br>Required |
| **Output** | Result STRING | Required |
| **Description** | This function applies a digital filter algorithm to a data file *Element* that corresponds to a classic Butterworth low-pass or high-pass filter. This digital filter does not shift the filtered data with respect to the unfiltered data (as do analog filters). If *Type* equals "lowpass", a low-pass Butterworth filter is emulated; if *Type* equals "highpass", a high-pass Butterworth filter is emulated. *Poles* specifies the number of poles of the filter. *Frequency* specifies the cut-off frequency.<br>This digital filters does NOT resolve alias-effects ! | |
| **Example(s)** | ["Result\filtData"] = filter("Pos6D\ObjectA", "lowpass", 6, 5.0000) | |
| **See also** | | |

## 16.16. findmaximum( )

| | | |
|---|---|---|
| **Syntax** | [Output] = findmaximum (Element, ExtraComment) | |
| **Input** | Element STRING<br>ExtraComment STRING | Required<br>Optional |
| **Output** | Output STRING | Required |
| **Description** | This function finds the maximum for each column in the data file *Element* . These values are appended as one line with multiple columns to the element *Output* (if *Output* doesn't exist, it is created). If *ExtraComment* is specified, an extra column is added, holding this comment string, serving as markers for 2D-graphs. If *ExtraComment* is not specified, the name of *Element* is used. | |
| **Example(s)** | ["Result\maxi"] = findmaximum("acquisition\camera\test01")<br>["Result\maxi"] = findmaximum("acquisition\camera\test01", "this was my test") | |
| **See also** | findminimum( ) | |

## 16.17. findminimum( )

| | | |
|---|---|---|
| **Syntax** | [Output] = findminimum (Element, ExtraComment) | |
| **Input** | Element STRING<br>ExtraComment STRING | Required<br>Optional |
| **Output** | Output STRING | Required |
| **Description** | This function finds the minimum for each column in the data file *Element* . These values are appended as one line with multiple columns to the element *Output* (if *Output* doesn't exist, it is created). If *ExtraComment* is specified, an extra column is added, holding this comment string, serving as markers for 2D-graphs. If *ExtraComment* is not specified, the name of *Element* is used. | |
| **Example(s)** | ["Result\maxi"] = findminimum("acquisition\camera\test01")<br>["Result\maxi"] = findminimum("acquisition\camera\test01", "this was my test") | |
| **See also** | findmaximum( ) | |

## 16.18. globaladd( )

| | | |
|---|---|---|
| **Syntax** | globaladd (Element1, …) | |
| **Input** | Element1 STRING | Required |
| **Output** | | |
| **Description** | This function adds one or more elements to the global elements list. Global elements are elements that, when used as script input parameter, do not need to be prepended with the test name. This is very usefull for identifications, … | |
| **Example(s)** | globaladd("Identification\StaticFrame")<br>globaladd("Identification\StaticFrame", "Identification\ObjectA") | |
| **See also** | globalemptylist( ) | |

## 16.19. globalemptylist( )

| | | |
|---|---|---|
| **Syntax** | Globalemptylist ( ) | |
| **Input** | | |
| **Output** | | |
| **Description** | This function makes the global elements list empty. | |
| **Example(s)** | globalemptylist( ) | |
| **See also** | globaladd( ) | |

## 16.20. identify( )

| | | |
|---|---|---|
| **Syntax** | identify (ScriptName) | |
| **Input** | ScriptName STRING | Optional |
| **Output** | | |
| **Description** | This function starts the application, used to identify frames or reference frames. In most cases, this application will be GeoLoc. The parameter *ScriptName* is the full path and filename of a GeoLoc script: if *ScriptName* is specified, this script will be automatically executed. | |
| **Example(s)** | identify( )<br>identify("C:\GeoLoc\scripts\measureReferenceFrame.script") | |
| **See also** | | |

## 16.21. increasetestname( )

| | | |
|---|---|---|
| **Syntax** | increasetestname ( ) | |
| **Input** | | |
| **Output** | | |
| **Description** | If a test name was specified before ( see chapter 17.1 ), this function will increment the test name with 1: if the specified test level was **myTest**, increasetestname( ) will create a new test level called **myTest01**. If called once more, increasetestname( ) will create a new test level called **myTest02**. This way, the same script can be re-used without changing element names, ànd without overwriting previously measured files. | |
| **Example(s)** | settestname("myTest") % here, the first test level is specified<br>increasetestname( ) | |
| **See also** | settestname( ) | |

## 16.22. joinmeasurement( )

| | | |
|---|---|---|
| **Syntax** | [Result] = joinmeasurement(Element1, FirstLed1, nofLeds1, Element2, FirstLed2, nofLeds2) | |
| **Input** | Element1 STRING<br>FirstLed1 INTEGER<br>nofLeds1 INTEGER<br>Element2 STRING<br>FirstLed2 INTEGER<br>NofLeds2 INTEGER | Required<br>Required<br>Required<br>Required<br>Required<br>Required |
| **Output** | Result STRING | Required |
| **Description** | This function joins 2 measurement files into one. This allows a user to merge two measurement files, measured on the same object but with two cameras with complementary field-of-views. The function will create the measurement file *Result* by taking *nofLeds1* from element *Element1* , starting at led *FirstLed1*; next, *nofLeds2* from element *Element2* starting from led *FirstLed2* are pasted as new columns to the right.<br>The function assumes that *Element1* and *Element2* are both expressed in a common reference frame. Before joining, a check is performed on the name of the reference of each measurement file: if the names differ, a warning is issued (but still the join will be performed: because it is possible that the user has given different names to the reference, but that the reference is physically equal). | |
| **Example(s)** | ["acq\Pos3D\object"] = joinmeasurement( "acq\Pos3D\object_left", 1, 3, "acq\Pos3D\object_right", 1, 3) | |
| **See also** | | |

## 16.23. message( )

| | | |
|---|---|---|
| **Syntax** | Message(Message) | |
| **Input** | Message STRING, FLOAT or INTEGER | Required |
| **Output** | | |
| **Description** | This function displays the string *Message* in the DMM message window. | |
| **Example(s)** | message("This is an informational message")<br>message(5.023)<br>message(5)<br>message(myVariable) | |
| **See also** | waitmessage( ), error( ), warning( ) | |

## 16.24. offset( )

| | | | |
|---|---|---|---|
| **Syntax** | [Result] = offset (Element, Type ) | | |
| **Input** | Element<br>Type | STRING<br>STRING CONSTANT {“START”, ”END”, ”MEAN”} or STRING | Required<br>Required |
| **Output** | Result | STRING | Required |
| **Description** | This function creates *Result* which is equal to *Element* , but with an offset for each channel. The type of offset is dictated by the parameter *Type*. If *Type* equals the constant string “START”, from each column the average of the 30 first positions will be subtracted (so in *Result*, each column starts at 0). If *Type* equals the constant string “END”, from each column the average of the 30 last positions is subtracted (so in *Result*, each column ends at 0). If *Type* equals the constant string “MEAN”, from each column the mean of the entire column is subtracted (so in *Result*, the average of each column is).<br>If *Type* equals the name of an existent element, for each value in *Result* the following applies: *Result*(i,,j) = *Element*(i,,j) - *Type*(i,,j) . | | |
| **Example(s)** | [“offset\motion”] = offset(“Pos6D\motion”, “START”)<br>[“offset\objA_B”] = offset(“Pos6D\objectA”, “Pos6D\objectB”) | | |
| **See also** | | | |

## 16.25. orientationchange( )

| | | | |
|---|---|---|---|
| **Syntax** | [Result] = orientationchange (Element, NewConvention ) | | |
| **Input** | Element<br>NewConvention | STRING<br>STRING CONSTANT {“rpy”, ”scs”, ”skrew”} | Required<br>Required |
| **Output** | Result | STRING | Required |
| **Description** | For 6D frames, the calculated angles are expressed using a default angular convention (see chapter 14.3.1). This function will express a 6D frame *Element* using a chosen angular convention. The new angular convention depends on the value of *NewConvention* : if *NewConvention* equals “rpy”, the new convention will be Roll-Pitch-Yaw. if *NewConvention* equals “scs”, the new convention will be Steer-Camber-Spin. if *NewConvention* equals “skrew”, the new convention will be the screw vector (independent RX, RY, RZ rotations).<br>See chapter “Orientation conventions” on page 19 for definitions and usage of the different angular conventions. | | |
| **Example(s)** | [“pos6D\objRPY”] = orientationchange(“Pos6D\objSCS”, “rpy”) | | |
| **See also** | | | |

## 16.26. overshoot( )

| | | | |
|---|---|---|---|
| **Syntax** | [Output,TimeSeries] = overshoot (Element, Axis, Comment ) | | |
| **Input** | Element<br>Axis<br>Comment | STRING<br>STRING CONSTANT {“XYZA”}<br>STRING | Required<br>Required<br>Optional |
| **Output** | Output<br>TimeSeries | STRING<br>STRING | Required<br>Optional |
| **Description** | This function calculates the overshoot for each channel in *Element* . Overshoot is defined as the difference between the maximal channel excitation and the mean end value. Overshoot( ) will add a new line. These overshoot values are appended as one line with multiple columns at the end of element *Output* (if *Output* doesn’t exist, it is created).<br>The parameter *Axis* determines in which directions the overshoot is calculated: if *Axis* contains “X”, “Y” or “Z”, the overshoot according to the X-, Y- and/or Z-axis of the reference frame is calculated. if *Axis* contains “A”, the absolute value of the overshoot ( $\|D\| = ?(?x^2 + ?y^2 + ?z^2)$ ) is calculated.<br>If *Comment* is specified, an extra column is added, holding this comment string, serving as markers for 2D-graphs. If *Comment* is not specified, the name of *Element* is used.<br>If *TimeSeries* is specified, a “normalized overshoot” time serie is created (always positive overshoot, end value offsetted to zero). | | |
| **Example(s)** | [“ovsh\slam”] = overshoot(“Pos6D\doorslam01”, “X”)<br>[“ovsh\slam”] = overshoot(“Pos6D\doorslam02”, “XZ”, ”2nd test”)<br>[“ovsh\slam”] = overshoot(“Pos6D\doorslam03”, “XYA”, ”3rd test”)<br>[“ovsh\slam”,”ovsh\time\Slam04]=overshoot(“Pos6D\doorslam04”, “XYZA”, ”4th test”) | | |
| **See also** | | | |

## 16.27. pointinframe( )

| | | | |
|---|---|---|---|
| **Syntax** | [Result] = pointinframe (Frame, Points) | | |
| **Input** | Frame<br>Points | STRING<br>STRING | Required<br>Required |
| **Output** | Result | STRING | Required |
| **Description** | The element *Frame* expresses the motion & orientation of the origin of a dynamic frame with respect to a reference frame. However, if you are interested in the motion of another point of the dynamic frame with respect to the reference frame, you can use this function to calculate it.<br>The element *Points* is a matrix, holding XYZ-coordinates of the points-of-interest, expressed in the dynamic frame. *Result* will hold the 3D-motions of each point-of-interest with respect to the reference frame in which the dynamic frame *Frame* is expressed. | | |
| **Example(s)** | [“Pos3D\engineCorners”] = pointinframe( “Pos6D\engineVsWorld”, ”Identification\CornersVsEngine” ) | | |
| **See also** | | | |

## 16.28. polyfit( )

| | | | |
|---|---|---|---|
| **Syntax** | [Result, Residu] = polyfit (Xinput, Xchannel, Yinput, Ychannel, Degree) | | |
| **Input** | Xinput<br>Xchannel<br>Yinput<br>Ychannel<br>Degree | STRING<br>INTEGER<br>STRING<br>INTEGER<br>INTEGER | Required<br>Required<br>Required<br>Required<br>Required |
| **Output** | Result<br>Residu | STRING<br>STRING | Required<br>Optional |
| **Description** | This function fits a polynome function of a measured or calculated timeseries. The polynome can also be calculated to express two channels versus each other (e.g. the linear regression between force and displacement).<br>The polynome fit of *Degree*–th order will use the *Xchannel*-th channel of element *Xinput* as X-coordinates, and the *Ychannel*-th channel of element *Yinput* as Y-coordinates as input data. The polynome X-values (equal to *Xchannel*-th channel of element *Xinput)* and calculated Y-values will be saved in *Result*. If *Residu* is specified, *Residu* will contain the errors between the original data and the polynome fit. | | |
| **Example(s)** | [“linearRegression”] = polyfit(“Pos3D\LedDisplacement”, 3, “Analog\twoForces”, 2, 1 )<br>[“cubicRegression”, “Err”] = polyfit(“Pos3D\A”, 1, “Pos3D\B”, 1, 3 ) | | |
| **See also** | | | |

## 16.29. rename( )

| | | | |
|---|---|---|---|
| **Syntax** | rename (OldName, NewName) | | |
| **Input** | OldName<br>NewName | STRING<br>STRING | Required<br>Required |
| **Output** | | | |
| **Description** | This function renames an existing element *OldName* to an element called *NewName.* | | |
| **Example(s)** | rename(“Position6D\ObjectImReff”, “Position6D\ObjectInRef”) | | |
| **See also** | | | |

## 16.30. saveall( )

| | |
|---|---|
| **Syntax** | saveall ( ) |
| **Input** | |
| **Output** | |
| **Description** | This function saves all elements to disk the moment this function is called. This prevents elements, that are still kept in memory, to be lost should an unexpected power failure or system crash occur. |
| **Example(s)** | saveall( ) |
| **See also** | |

## 16.31. settestname( )

| | | |
|---|---|---|
| **Syntax** | settestname (newTestName) | |
| **Input** | NewTestName STRING | Required |
| **Output** | | |
| **Description** | This function sets the test name ( see chapter 17.1 ), that is to be used from now on, to *NewTestName*. If an empty string is specified, the root level becomes the active level again. | |
| **Example(s)** | Settestname(“testBeforeModifications”)<br>Settestname(“”) | |
| **See also** | increasetestname( ) | |

## 16.32. terminate( )

| | | |
|---|---|---|
| **Syntax** | terminate ( ) | |
| **Input** | | |
| **Output** | | |
| **Description** | This function terminates DMM. | |
| **Example(s)** | terminate( ) | |
| **See also** | | |

## 16.33. transform( )

| | | |
|---|---|---|
| **Syntax** | [Result] = transform (Element, NewRef, LedNumber ) | |
| **Input** | Element STRING<br>NewRef STRING<br>LedNumber INTEGER | Required<br>Required<br>Optional |
| **Output** | Result STRING | Required |
| **Description** | This function transforms (that is, changes the frame of reference) the element *Element* into a new reference frame called *NewRef* . The result will be stored in a new element called *Result* . If *NewRef* is a static frame or a dynamic frame element, the transformation will be a 6D-transformation (that is, the orientation are converted to the new reference as well). If *NewRef* is a LED file, and a valid led sequence number is specified, the transformation will be a 3D-transformation: this means that led no. *LedNumber* will be the origin for the transformation, but that the 3 orientation angles will not change (this boils down to a pure translation of the old frame of reference). | |
| **Example(s)** | [“Pos6D\ObjInNewRef”] = transform(“Pos6D\ObjInOldRef” , ”Identification\NewRef)<br>[“Pos6D\ObjInDynFrame”] = transform(“Pos6D\ObjInOldRef” , ”Pos6D\DynFrameInOldRef)<br>[“Pos6D\ObjInNewRef”] = transform(“Pos6D\ObjInOldRef” , ”Acquisition\Position3D\LedsnOldRef”, 3 ) | |
| **See also** | | |

## 16.34. velocity( )

| | | |
|---|---|---|
| **Syntax** | [Result] = velocity (Element, Scalar ) | |
| **Input** | Element STRING<br>Scalar STRING CONSTANT {"scalar"}<br>LedNumber INTEGER | Required<br>Optional |
| **Output** | Result STRING | Required |
| **Description** | This function derives any element *Element* (holding timeseries) once versus time. The velocities will be expresses separately along the axes of the frame of reference. If *Scalar* is defined, the norm of the total velocity ( $\|V\| = \sqrt{(V_X^2 + V_Y^2 + V_Z^2)}$ ) is calculated. | |
| **Example(s)** | ["Velocity\ObjSpeed"] = velocity("Position6D\Obj")<br>["Velocity\ObjSpeed"] = velocity("Position6D\Obj", "scalar") | |
| **See also** | | |

## 16.35. wait( )

| | | |
|---|---|---|
| **Syntax** | wait ( WaitTime ) | |
| **Input** | WaitTime INTEGER | Optional |
| **Output** | | |
| **Description** | This function stalls script execution for *WaitTime* milliseconds. If *WaitTime* is not specified, the software waits for a key to be pressed. | |
| **Example(s)** | wait( )<br>wait(5000) | |
| **See also** | | |

## 16.36. waitmessage( )

| | | |
|---|---|---|
| **Syntax** | waitmessage ( Message ) | |
| **Input** | Message STRING | Required |
| **Output** | | |
| **Description** | This function shows a pop-up box with a message string *Message* , which the user has to confirm by clicking OK. | |
| **Example(s)** | waitmessage("Click the OK-button to proceed ) | |
| **See also** | | |

## 16.37. warning( )

| | | |
|---|---|---|
| **Syntax** | warning(WarnMessage) | |
| **Input** | WarnMessage STRING, FLOAT or INTEGER | Required |
| **Output** | | |
| **Description** | This function generates a warning condition that will display the string *WarnMessage* in the DMM message window. **Be carefull !** Any action, associated with warning conditions, will be executed as well ! (e.g. setting bits of the synchro module high, …) | |
| **Example(s)** | warning("This is an error message")<br>warning (5.023)<br>warning (5)<br>warning (myVariable) | |
| **See also** | message( ), error( ), waitmessage( ) | |

# 17. Data organization

Just like its precessor, DMM Modular allows the user to store his data in a more hierarchic organisation – and in an even better way.
DMM Modular still orders data in projects, to separate identifications, measurement files and calculations between independent measurements. For elaborate testing schemes, sometimes the overview of the data was lost. Since DMM Modular 2.1, the software introduces one more level : the “test” level.

## 17.1. The ‘test’ level

On a test level, the user can group all files that belong together on a very local scope ; on a project level, the user can group measurements that have got little to do with each other.

Possible project organisation schemes are :
- Object-based : for each new object to measure, a new project is created. For this object, some standard test can be defined
- Person-based : every operator can have a project, in which he can create new tests as he wishes

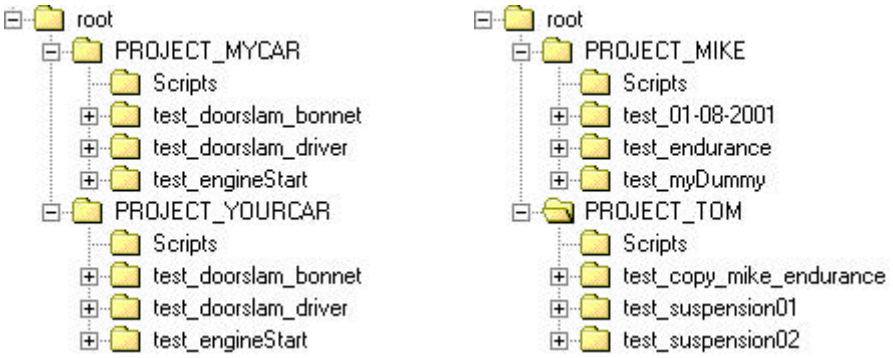
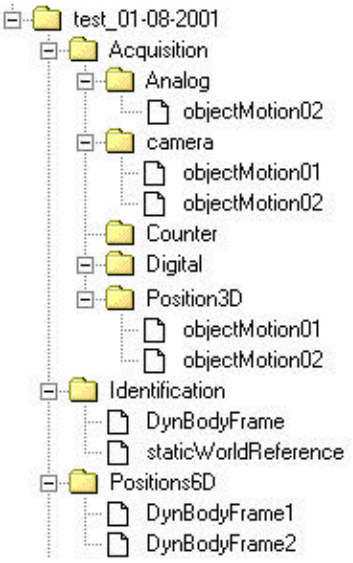



*Figure 51: data organisation: object-based (left) or person-based (right)*

Of course, the list above does not restrict you to the listed organisation schemes. It is up to the user to define a scheme that fits your needs and/or company policy best.

For each test, a new hierarchy is created to separate identifications, measurement files, 6D-files and other calculated files.
The different folders are:




**Identification:** contains identifications of reference frames ans objects ( see chapter “Identifying objects” on page 25.
**Acquisition:** contains all raw acquisition files:
**Camera:** raw individual LED 3D data in camera frame.
**Position3D:** individual LED 3D data in world reference frame
**Analog:** ADC600 analog input measurements
**Digital:** ADC600 digital input measurements
**Counters:** ADC600 encoder/counter input measurements
**Position6D:** contains calculated object-in-world 6D motion files

It is possible that other folders exist, since they can be

Freely specified when calculating data (e.g. to save all velocity calculations in a folder “test_01-08-2001\velocity” )

To create a new project, see chapter “‘Project’ menu” on page 60.

To create a new test within a project, select the option **“set test name”** from the project menu.
A pop-up form appears in which the name of the new test can be entered.
If the test already existed, this test will be the active test untill a new test is created, or untill the software is closed.
If the test did not already exist, a new test is created. The test folder will NOT appear in the workspace window untill the first data file is created or imported !

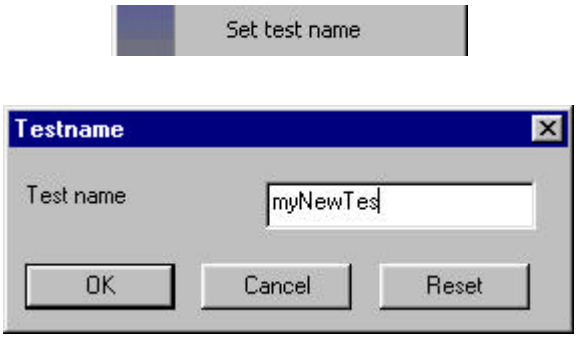



*Figure 52: data organisation: ‘set test name'*

In case the user wants to execute multiple runs of the same test, DMM Modular has a feature to increment a test. To when incrementing a test, DMM Modular will create new tests, whos names end with sequence numbers: “myNewTest1”, “myNewTest2”, …

Incrementing test names can only be done from within a script:

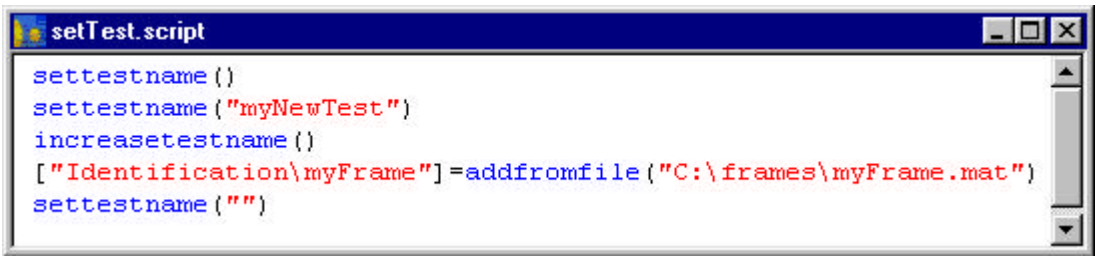



*Figure 53: data organisation: incrementing test names from a script*

**Settestname( )** will create/open a new project with the same pop-up form as when creating/opening a test from the menu.

**Settestname(<new Test Name>)** will create a new test without the pop-up form.

**Increasetestname( )** will increment the last opened test name (in the example above, a new test called “myNewTest1” will be created) and set this test to be the active one.

The fourth line will add a frame in the myNewTest1\Identification-folder.

**Settestname(“”)** (empty string) will make the project level the active one: all files created will be added in the hierarchical tree of the project and not in one of the tests within that project.

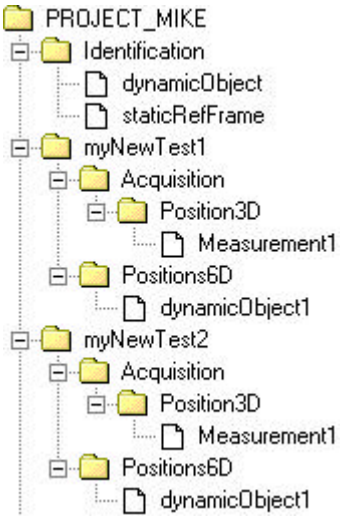
If the original name ends with a character (e.g. “myNewTest”), the incremented test names will start with 1 ( “myNewTest1”, “myNewTest2”, …)
If the original name ends with a digit (e.g. “myNewTest1”), the incremented test names will start with the next digit ( “myNewTest2”, “myNewTest3”, …). One exception: if the digit has a “leading zero” ( e.g. “ABC01”), this is not considered to be a digit, and omitted; the incremented test name will start from 1 ( thus “ABC1”, “ABC2”, … )

## 17.2. Global variables

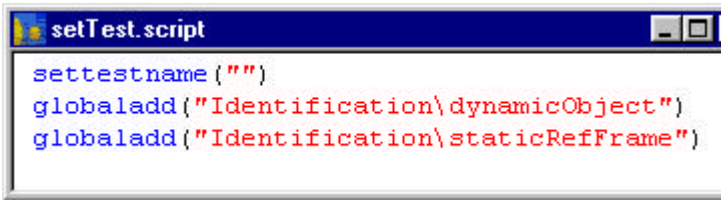
In order to make tests entirely independent from each other, each test has its own hierarchical folder structure and are therefore shielded from one another. But sometimes,

you want some global files ( e.g. identifications of static and dynamic frames ) to be used in all tests; therefore, we need a mechanism to make some data files **global**, that is, visible to all tests.

Assume the following situation: we make a single identification of the dynamic frame “dynamicObject” and a static frame “staticRefFrame”, and we want to re-use these in two tests, “myNewTest1” and “myNewTest2”. In each of these tests, we need the identification frames to calculate the object with respect to the static frame.




The script function **globaladd( <element> )** adds <element> to a (invisible) search path:




Settestname(“”) makes sure our current level is the project level. The two next lines add the elements “dynamicObject” and “staticRefFrame” to the search path.

If we would now open test “myNewTest1”, acquire the measurement file “Measurement1”, and execute a 6D frame calculation based on the identification file “dynamicObject”, DMM Modular will first check if there is such an identification in the “PROJECT_MIKE\myNewTest1\Identification” folder, if not, he will search the global search path, and use the identification found in “PROJECT_MIKE\Identification”.

Should identification exist in the “PROJECT_MIKE\myNewTest2\Identification” folder with the same name, this identification would be discarded, because it was not added to the global search path.

If you want to empty a global search path, use the script function **globalemptylist( )**: this will erase all entries from the global search path.

**Notes:**

- The functions globaladd( ) and globalemptylist( ) are script functions: they can not be used from the main menu.
- The scope of the global search path is limited to the current project.
- We suggest to create one script per project that defines all global elements, and one script to erase them from the project. This script could have a fixed name in each project (e.g. “defineGlobals.script” and “eraseGlobals.script”), and should be saved in the <*project*>\Scripts sub-directory on the hard disk. It should be ran manually every time a project is opened. This is however merely a proposed convention: it is up to the user to set up a procedure that suits him best.
- The global search path ceases to exist when the software is closed ! When you are starting the software a second time, you must re-run a script that adds all required elements to the global search path.

# 18. Maintenance

This chapter describes the software maintenance for Rodym DMM Modular.

The guidelines described in this chapter are recommendations: they are not stringent, but not applying them might in the long run inhibit the easy and comfortable use of your system.

a. Make regular system backups, especially if you are re-using the same identifications over and over again.
b. empty your ‘Deleted items’ bin regularly. The ‘Deleted items’ are project-dependant: emptying it will not erase the ‘Deleted items’ of another project.
c. Once a project is closed, copy it onto a network drive. Should the system be down for a while, and you urgently need data from the system, you can check the network drive for the data.
d. When measuring with a script, let the script make a backup copy of your raw data: should you change the data by accident, e.g. in the spreadsheet view, you can retrieve the correct data from the safety copy.

# 19. Troubleshooting

This chapter describes the problems that are most likely to arise, and the way to solve them.
Should any problem not occur in the list, please notify your nearest Krypton representative.

## 19.1. General

?? **If I start the software, I get a crash immediately after I log in:**
- Probably your hardware key is not connected, or not connected properly.
- If this is not the problem, contact Krypton at support@krypton.be

?? **When I re-install my software, some menu items are missing:**
- Open the menu “Tools” – “Customize”. Go to the tab “menu” and click “Reset”.

?? **After a system crash, my latest files are missing from the workspace window:**
- The data file administration is corrupted. Open your project directory and use “Add from file” to manually import the data files again
- After each measurement or calculation or script run, save the workspace.

## 19.2. Identifying objects

?? **When I choose “Identify” – “Start”, nothing happens:**
- Check in the “Tools” – “Option” menu if your Geloc application is pointing to the correct application.
- Check that the IP port number is not assigned to a different task.

## 19.3. Measuring single LED’s or objects

?? **Acquisition will not start, or application crashes when I confirm the acquisition settings :**
- Make sure the camera is enabled ( checkbox upper-left on the camera-tab of the acquisition settings form )
- Check the camera settings with CamCheck
- In Rodym CMM, make sure the Space® Probe is stopped
- The acquisition frequency might be too high. Multiply the total number of LED’s with the acquisition frequency ; add 10% . The frequency you’ve just calculated should be equal or less than the frame frequency in CamCheck. If not, reduce the acquisition frequency untill it does, or increase the LED frequency (max. 3500 Hz). It is advised to activate the “Set automatic” option in CamCheck.

## 19.4. Adding calculation modules

?? **My toolbars and menu are garbled after installing a new module:**
- In the main menu, go to “Tools” – “customize”. Select the tab “Menu” , and click on the “Reset”-button.
- Select the tab “Toolbars” and click “Reset all”. See also chapter “‘Customize’” on page 65.

## 19.5. Creating spreadsheets & graph’s

?? **My cursors are not functioning:**
- Go to the “cursors”-tab in the output window. Right-click and choose “Synchronize”.

## 19.6. Exporting to Matlab

?? **When I close Matlab and open it again, I can't teleport my data files any more:**

- You've lost the connection between DMM Modular and Matlab. Save your workspace, close Matlab and DMM Modular, start Modular again. In DMM Modular, click on the Matlab icon.

?? **I can't teleport my data files, not even when DMM Modular started Matlab for the very first time. In the message window, it says: "Matlab could not be started"**

- Probably, DMM Modular could not "register" itsself with Matlab during installation. It is likely that Matlab was installed after DMM Modular was installed.
- Re-install DMM Modular
- If this does not work, delete the files LIBENG.DLL, LIBMX.DLL, LIBMI.DLL and LIBUT.DLL from the DMM <root>\bin-directory ( by default, it's the directory *c:\rodymdmm2\bin* ). Locate these files in the bin-directory of your currently installed Matlab-version. Copy them to the DMM <root>\bin-directory
- Open a command window, go to the bin-directory of your current Matlab-version and execute :
  *Matlab /regserver*

# 20. Glossary

**Alignment** : an alignment is a procedure in that establishes a relation between an object frame and the coordinate system of the measurement machine. The primary purpose of an alignment is to express measured coordinates into the coordinate system of the object. However, the math's behind it make it identical to measuring a *static frame*.

**Dynamic frame** : a dynamic frame is a *static frame* with LED's attached to it. It is the result of an *identification*. If Modular knows the position of the LED's in that static frame, and it sees the LED's floating somewhere in space, it can calculate the position and orientation of that frame with respect to the camera.

**Identification** : an identification is a procedure that establishes a unique link between an object and LED's that are attached to this object. It is a three-step procedure :

1) establish the *transformation* between the object frame and camera frame
2) measure the LED's on the object in camera frame
3) calculate the LED's from camera frame into the object frame with the *transformation*.

**Frame** : "Frame" can have two meanings :

1) when referring to a "frame", we usually refer to a coordinate system, a set of three orthogonal axes. Since we can assign a frame to each object we measure, an object is usually replaced by its frame. Each frame can be either the reference for a motion, or can be the object we want to express with respect to a reference.
2) "Frame" can also mean a snap-shot of all configured LED's. Before a measurement, we must configure the number of LED's we will measure. The camera will scan its measurement volume for LED 1, 2, …, n . Each set of n LED coordinates is also called a "frame".

**Object frame** : this is a frame that is explicitly linked to an object. No special actions must be taken to calculate an object frame.

**Object identification** : see *Identification.*

**Static frame** : a static frame is a *frame* whos position is fixed with respect to the reference frame. The relation between these two frames is called a "*transformation*".

**Transformation** : a transformation is a static relation between two *frames*. A transformation can express two mathematical features :

1) it expresses one frame into another reference frame, both in position as in rotation.
2) it allows to recalculate coordinates, known in one frame, into a second frame, or vice versa. This is usefull, e.g. when calculating individual LED positions from camera frame into a chosen world-frame.

# 21. Index

2D-graph ..... 11, 21, 50
3D ..... 15
3D-view ..... 11, 21, 53
6D ..... 15
acquisition ..... 24, 30, 32
acquisition menu ..... 66
Acquisition( ) ..... 74
Addfromfile( ) ..... 74
Advanced ..... *See* base
alignment ..... 93
base ..... 33
base menu ..... 67
calculate point in 6D frame ..... 39
Calculatemean( ) ..... 75
Calculatestd( ) ..... 75
camera ..... 11
change orientation convention ..... 38
Clearmemory( ) ..... 75
CMM ..... 10
coarse mode ..... 46, 47
Compensatelinearphasefilter( ) ..... 76
configure acquisition ..... 25
configure camera ..... 19
Configure DMM Modular ..... 22
Copy( ) ..... 76
copyright ..... 9
create 6D data ..... 32
createframe( ) ..... 76
CSV-file ..... 38, 39, 48, 60
cursors ..... 45, 91
data menu ..... 48
data organisation ..... 87
Delete ..... 49
delete( ) ..... 77
distance ..... 35
distance( ) ..... 77
DMM Modular ..... 10
dynamic frame ..... 16, 93
edit menu ..... 49
environment ..... 18
error( ) ..... 77
Excel ..... 47, 49, 52
export ..... 38, 39
export( ) ..... 78
FFT( ) ..... 78
filecopy( ) ..... 78
filter (digital) ..... 42
filter( ) ..... 79
findmaximum( ) ..... 79
findminimum( ) ..... 79
fit polynome ..... 40
Frame ..... 93
frequency spectrum ..... 42
function script ..... 70
GeoLoc ..... 10, 23, 58, 64
globaladd( ) ..... 80
globalemptylist( ) ..... 80, 89
graph's ..... 45, 91
help menu ..... 67
Identification ..... 93
identify menu ..... 64
identify( ) ..... 80
identifying objects ..... 23, 91
increasetestname( ) ..... 80
installation ..... 13
joinmeasurement( ) ..... 81
joinmeasurements ..... 43
LED visibility ..... 19, 30
legend ..... 52
markers ..... 48, 52
Matlab ..... 56, 92
Matlab function ..... 69
maximum ..... . See Statistics
mean ..... . *See* Statistics
measure ..... 32
measurement setup ..... *See* setup
measuring LED's ..... 24, 91
message( ) ..... 81
minimum ..... . *See* Statistics
modules ..... 11, 32, 44
object file ..... 21
Object frame ..... 93
Object identification ..... *See Identification*
offset ..... 34, 39
offset( ) ..... 82
orientation convention ..... 17, 38, 39
orientationchange( ) ..... 82
overshoot ..... 36, 41
overshoot( ) ..... 83
password ..... 20, 64
pointinframe( ) ..... 83
polyfit( ) ..... 84
polynomial ..... *See fit polynome*
project ..... *See data organisation*
Project menu ..... 57
properties ..... 21, 53
range ..... 49
rename( ) ..... 84
rule ..... 51
Run ..... 65
saveall( ) ..... 84
script menu ..... 64
scripting ..... 11, 68
search menu ..... 49
settestname( ) ..... 85
settings ..... 18
setup ..... 18
Space Probe ..... 19
spreadsheet ..... 11, 21, 91
Standard ..... *See* base
static frame ..... 16, 93
statistics ..... 37, 42

std deviation ..... *See* Statistics
teleport ..... 23, 56, 92
terminate( ) ..... 85
test ..... *See data organisation*
test level ..... *See data organisation*
tools ..... 59
trademarks ..... 9
transform ..... 33
transform( ) ..... 85
transformation ..... 11, 16, 93
user interface ..... 20
user login ..... 20, 64
velocity( ) ..... 86
view ..... 59
wait( ) ..... 86
waitmessage( ) ..... 86
warning( ) ..... 86
writing scripts ..... 68
zoom ..... 52